Protocolos de Controle de Congestionamento Amigáveis ao TCP

Henri Alves de Godoy

Trabalho Final de Mestrado Profissional

Instituto de Computação
Universidade Estadual de Campinas

Protocolos de Controle de Congestionamento Amigáveis ao TCP

Henri Alves de Godoy

Fevereiro de 2004

Banca Examinadora:

- Prof. Dr. Nelson L. S. da Fonseca (Orientador)
- Prof. Dr. Omar Branquinho
  Pontifícia Universidade Católica de Campinas – PUC-Campinas
- Prof. Dr. Edmundo R. M. Madeira
  Instituto de Computação - UNICAMP
- Prof. Dr. Ricardo Dahab (Suplente)
  Instituto de Computação – UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Godoy, Henri Alves de

G548p Protocolos de controle de congestionamento amigáveis ao TCP / Henri Alves de Godoy -- Campinas, [S.P. :s.n.], 2004.

Orientador : Nelson Luis Saldanha da Fonseca

Trabalho final (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação.

1. Protocolos de redes de computadores. 2. TCP/IP (Protocolo de rede de computação). 3. Internet (Redes de computação). I. Fonseca, Nelson Luis Saldanha da. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

PROTOCOLOS DE CONTROLE DE CONGESTIONAMENTO AMIGÁVEIS AO TCP

Este exemplar corresponde à redação final do Trabalho Final devidamente corrigida e defendida por Henri Alves de Godoy e aprovada pela Banca Examinadora.

Campinas, 27 de fevereiro de 2004.

Prof. Dr. Nelson L. S. da Fonseca
(Orientador)

Trabalho Final apresentado ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na área de Redes de Computadores.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 27 de fevereiro de 2004, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. Ricardo Dahab
IC - UNICAMP

Prof. Dr. Edmundo Roberto Mauro Madeira
IC - UNICAMP

Prof. Dr. Nelson Luis Saldanha da Fonseca
IC-UNICAMP

Resumo

O oferecimento crescente dos serviços de banda larga tem tornado cada vez mais comum o uso de aplicações multimídia na Internet, e o número de transmissões de áudio e vídeo tem crescido nos dias atuais. No entanto, tais fluxos tipicamente utilizam o protocolo de transporte UDP, que se comporta de forma injusta em relação ao TCP.

As conexões TCP procuram sempre se adaptar às condições da rede, compartilhando a largura de banda de maneira justa, aproveitando a banda disponível. As aplicações baseadas em UDP, quando competem com tráfegos TCP, tendem a ocupar toda a banda disponível, pois não reagem a nenhuma notificação de congestionamento da rede.

O presente trabalho visa apresentar mecanismos de controle de congestionamento e regras para calcular o nível de congestionamento da rede; pretende, ainda, adaptar os fluxos UDP para coexistirem com as conexões TCP, transformando o tráfego UDP das aplicações multimídia em amigáveis ao TCP, tentando evitar, desse modo, um colapso na rede e a inanição do tráfego TCP.

Abstract

The increasing number of multimedia applications over the Internet typically uses UDP as transport protocol. UDP does not implement any congestion control mechanism and, thus, does not reduce its sending rate in a congested network, making unfair the share of bandwidth with TCP flow.

The present work presents existing mechanisms for adapting UDP senders to share the medium with TCP senders in a fair way. These mechanisms are called TCP friendly and they aim at avoiding network collapse and TCP starvation.

Agradecimentos

Agradeço a Deus por te me dado à vida e permitir a realização deste trabalho.

À minha esposa, Angela, pelo incentivo e apoio durante todo o tempo, não me deixando desanimar.

Aos meus pais, Hélio e Silvia, pela formação e educação que me deram.

Ao Professor Doutor, Nelson L. S. da Fonseca, pela dedicação na orientação deste trabalho.

Aos colegas do Instituto de Computação.

Sumário

Índice de Figuras

Índice de Tabelas

1. Introdução:

A crescente popularidade da Internet tem causado um aumento na demanda de tráfego e o seu sucesso é devido ao uso do protocolo *Internet Protocol* (IP) responsável pela transferência de dados, independente da tecnologia da camada de enlace. Junto com o protocolo de transporte *Transmission Control Protocol* (TCP), a pilha de protocolos TCP/IP é classificada como dominante na Internet, apropriado para aplicações onde se necessita uma transferência confiável de dados. Outro protocolo de transporte, o UDP (*User Datagram Protocol*), permite às aplicações utilizarem o serviço de entrega não confiável de datagramas, provido pelo IP.

O uso do mecanismo de controle de congestionamento do TCP aplicado ao modelo de melhor esforço (*best-effort*) da Internet tem como objetivo prevenir a ocorrência de congestionamento na rede e, além de proporcionar uma entrega confiável de dados, prover o controle de fluxo e de congestionamento na rede. Uma outra característica do mecanismo de controle de congestionamento do TCP é que as conexões TCP existentes compartilham, igualmente, a largura de banda disponível. Cada estação de um sistema final (*endhost*), onde reside o protocolo de transporte, controla sua taxa de transmissão, alterando o tamanho da janela de congestionamento do TCP em resposta a uma notificação de congestionamento inferida.

No entanto, nem todas as aplicações para transmissão de áudio e vídeo [REALNETWORKS 2000] utilizam, como protocolo de transporte, o TCP e, portanto, não praticam o mesmo conceito de um compartilhamento justo de largura de banda. Essas aplicações, tipicamente, não obedecem a nenhuma notificação de controle de congestionamento.

O oferecimento crescente dos serviços de banda larga tem tornado cada vez mais comum o uso de aplicações multimídia na Internet e o número de transmissões de áudio e vídeo tem crescido nos dias atuais. Tais fluxos tipicamente utilizam o protocolo de transporte UDP, que se comporta de forma injusta em relação ao TCP, pois os fluxos TCP reduzem sua taxa de transmissão de dados numa tentativa de eliminar o congestionamento, enquanto os fluxos UDP continuam a transmitir a taxa desejável. [FLOYD *et al*, 1999].

Faz-se necessário, portanto, definir mecanismos de controle de congestionamento e regras para calcular o nível de congestionamento da rede e adaptar os fluxos UDP para coexistirem com as conexões TCP, transformando o tráfego UDP das aplicações multimídia em amigáveis ao TCP, tentando evitar, assim, um colapso na rede e a inanição do tráfego TCP. [FLOYD 2000]

Neste trabalho, descrevem-se os protocolos amigáveis ao TCP. Os mecanismos de controle de congestionamento dos protocolos amigáveis ao TCP, existentes na literatura, são comparados e sua eficiência é apresentada, quando colocada em conjunto com diversos tráfegos TCP em ambientes heterogêneos. No Capítulo 2, é apresentado resumidamente o funcionamento do mecanismo de controle de congestionamento do protocolo TCP e a pilha TCP/IP. A definição do conceito amigável ao TCP e o método de classificação dos protocolos são apresentados no Capítulo 3. No Capítulo 4, estão os protocolos amigáveis ao TCP, suas características principais e seu funcionamento. Uma comparação entre os protocolos é demonstrada no Capítulo 5 e as considerações finais deste trabalho são apresentados no Capítulo 6.

2. Mecanismo de controle de congestionamento do TCP.

Congestionamento é a situação na qual existe uma quantidade de pacotes a serem transmitidos, cujo volume é maior do que a rede é capaz de suportar.. Quando há congestionamento existe um aumento no retardo e perdas. As retransmissões, devido às perdas, produzem um aumento no tráfego, podendo levar a rede a um colapso. Ao detectar o congestionamento incipiente, o TCP reduz a sua taxa de transmissão, utilizando os algoritmos [KUROSE 2002] de controle de congestionamento: *slow start, congestion avoidance, fast recovery e fast retransmit*.

Nas primeiras implementações do TCP, os transmissores começavam a transmissão de múltiplos segmentos até o tamanho da janela anunciada pelo receptor. Quando essas transmissões saíram do âmbito local e surgiram os roteadores e enlaces lentos entre o transmissor e o receptor, notou-se uma redução no desempenho da conexão TCP.

O algoritmo *slow start* acrescenta uma outra janela situada no transmissor, chamada de janela de congestionamento (*cwnd*). Quando uma conexão é estabelecida, a janela de congestionamento é iniciada em 1 segmento. A cada reconhecimento recebido, a janela de

congestionamento é incrementada por 1 segmento. O transmissor pode transmitir até o mínimo, entre a janela de congestionamento e a janela anunciada pelo receptor.

O transmissor inicia transmitindo 1 segmento e aguarda por seu reconhecimento (ACK). Ao receber esse reconhecimento, sua janela de congestionamento é incrementada de 1 para 2 e, agora, 2 segmentos podem ser enviados. Quando cada um dos 2 segmentos for reconhecido, a janela de congestionamento é incrementada para 4 (aumento exponencial).

Em algum ponto, a capacidade limite da rede é alcançada e os pacotes, ao longo do caminho, começam a ser descartados. Isso indica que a janela de congestionamento está grande. O *slow start* mantém em uma variável chamada *ssthresh (slow start threshold)* um limite para o crescimento exponencial do *slow start*. Se o valor do crescimento *cwnd* ultrapassar o valor da variável *ssthresh*, o modo de controle de fluxo do TCP é alterado de *slow start* para *congestion avoidance*.

Na fase de *congestion avoidance* [STEVENS 2000], a janela de congestionamento não aumenta exponencialmente, mas sim linearmente. A cada reconhecimento de todos os segmentos da janela, o tamanho da janela é incrementado de uma unidade.

Quando o congestionamento ocorre, indicado pela expiração do intervalo de temporização ou pela chegada de reconhecimentos duplicados, metade do tamanho atual da janela de congestionamento é armazenada na variável *ssthresh*. Se a indicação de congestionamento foi causada por uma expiração do intervalo de temporização, a janela de congestionamento *cwnd* é ajustada em 1 segmento.

A maneira como a janela de congestionamento é aumentada, a cada reconhecimento de novos dados pelo destino, depende do modo de controle de fluxo que está sendo adotado no momento: *slow start* ou *congestion avoidance*. Se a janela de congestionamento for menor ou igual ao valor mantido na variável *ssthresh*, o TCP encontra-se na fase de *slow start*, caso contrário está sendo realizado o *congestion avoidance*.

Modificações no algoritmo de *congestion avoidance* [JACOBSON 1990] permitiram a retransmissão prematura de um segmento perdido antes da expiração do intervalo de temporização, chamada de *fast retransmit*. Isso acontece quando três ou mais pacotes de reconhecimento duplicados são recebidos em seqüência, sinalizando que o segmento foi perdido.

Por fim, o algoritmo *fast recovery* realiza o cancelamento da fase de *slow start* do TCP, quando o algoritmo de *fast retransmit* sinaliza um congestionamento. Uma redução brusca do fluxo, utilizando o *slow start* não é aconselhável, pois dados ainda podem estar trafegando entre dois nós, o que pode agravar a situação de congestionamento.

2.1 O modelo de pilha do TCP/IP.

O modelo TCP/IP [COMER 1997] foi projetado, segundo uma arquitetura de pilha de 4 camadas conceituais que interagem com uma camada acima ou abaixo. A figura 1 ilustra as 4 camadas do modelo TCP/IP.

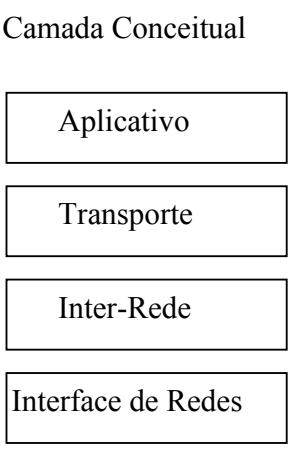


Figura 1. As quatro camadas conceituais do modelo TCP/IP.

- Camada de aplicativos: nível mais alto, responsável por programas aplicativos dos usuários, utilizando, cada um, seu próprio protocolo de comunicação (HTTP, FTP, SMTP, TELNET) que interagem com a camada de transporte abaixo, para enviar ou para receber dados.

- Camada de transporte: camada responsável pela comunicação de um programa aplicativo para outro chamado de fim-a-fim. Os protocolos de transporte TCP e UDP atribuem a cada programa um número de porta, de modo a saber onde entregar cada mensagem recebida pela rede. A camada de transporte tem, também, a função de dividir o fluxo de dados em pequenas partes (pacotes) para serem transmitidos para a camada seguinte.
- Camada Inter-Rede: camada responsável pelas decisões de roteamento de um datagrama. A entrega de um datagrama pode ser local ou para um roteador que passa o datagrama a uma interface de rede apropriada, podendo, assim, realizar a conexão entre redes. A camada Inter-Rede é responsável, também, pelo envio de mensagens de controle de erro ICMP (*Internet Control Message Protocol*).
- Camada da interface de rede: nível mais baixo, responsável pelas transmissões das informações de um *host* para o outro, em uma mesma mídia de acesso (Ethernet, PPP, SLIP).

3. Controle de congestionamento amigável ao TCP

As conexões TCP procuram sempre se adaptar às condições da rede, compartilhando a largura de banda de maneira justa, aproveitando a banda disponível. Muitas aplicações multimídia utilizam o UDP, tentando evitar as mudanças bruscas que acontecem na taxa de transmissão, sempre que a rede estiver congestionada, dada a necessidade de manter uma taxa constante para as transmissões de áudio e vídeo. As aplicações baseadas em UDP, quando competem com tráfegos TCP, tendem a ocupar toda a banda disponível, pois não reagem a nenhuma notificação de congestionamento da rede. [HASEGAWA *et al*, 2001]

Os resultados obtidos do compartilhamento de banda, em um experimento realizado entre uma transmissão UDP a uma taxa de 100 Mbps e nove conexões TCP, mostraram, claramente, que a transmissão UDP manteve sua taxa de transmissão constante em torno dos 100 Mbps, enquanto as conexões TCP permaneceram entre 0 a 20 Mbps. A maioria das aplicações multimídia possui mecanismos de adaptação de taxa proprietários, que objetivam oferecer uma melhor qualidade na sua apresentação.

Diante disso, é necessária a criação de novos protocolos classificados como amigáveis ao TCP, os quais buscam diminuir e equilibrar esse grau de injustiça, criado quando tráfegos UDP e TCP compartilham a mesma banda passante. Um fluxo é definido como sendo amigável ao TCP, quando a quantidade de dados transferidos ao longo do período não excede a quantidade de dados transferidos de uma conexão TCP sob as mesmas condições, segundo Widner [WIDNER *et al*, 2001]. Para alcançar esse controle, os fluxos UDP avaliam a velocidade de transmissão de uma conexão TCP (*throughput*), executando o controle amigável ao TCP através da observação de parâmetros como, por exemplo, taxa de perda e tempo de ida e volta (*Round Trip Time*). Esse controle ajusta a taxa de envio de pacotes.

Os protocolos amigáveis ao TCP podem ser classificados em relação a algumas características definidas por Widner [WIDNER *et al*, 2001]. Primeiramente, eles podem ser classificados como sendo baseados em janelas ou baseados na taxa de transmissão. Os protocolos que pertencem à categoria de janelas utilizam uma janela de congestionamento no transmissor ou no receptor, para garantir o comportamento amigável ao TCP. A janela de transmissão aumenta na ausência da indicação de congestionamento ou diminui na ocorrência de congestionamento. Já os protocolos baseados em taxa adaptam, dinamicamente, a taxa de transmissão de acordo com o retorno (*feedback*) da rede. Para alcançar o comportamento amigável ao TCP, alguns protocolos dessa categoria imitam o modelo AIMD (*Additive Increase Multiplicative Decrease*) enquanto outros ajustam sua taxa, de acordo com o modelo de tráfego do TCP.

Outra taxonomia de classificação leva em consideração o número de receptores, ou seja, se *unicast* ou *multicast*. Para transmissões *multicast* pode-se aplicar outro critério de classificação dos protocolos em relação à taxa de transmissão: de forma única (*single-rate*) ou em taxas variadas (*multi-rate*). A transmissão em taxas variadas permite ao controle de congestionamento uma alocação de banda mais flexível, ao longo dos diferentes caminhos, adaptando-se melhor às condições da rede.

4. Estudo dos protocolos amigáveis ao TCP.

Neste capítulo, os protocolos amigáveis ao TCP são apresentados, destacando-se os seus algoritmos de controle de congestionamento e vários aspectos que envolvem sua implementação. O desempenho dos protocolos, quando em situação de competição com tráfegos TCP, é mostrado através da reprodução de resultados existentes na literatura.

4.1 Protocolos Unicast Amigáveis ao TCP.

Os protocolos *unicast* amigáveis ao TCP são geralmente utilizados em aplicações multimídia ponto a ponto. Tanto o transmissor como o receptor conhecem um ao outro, e adaptam a taxa de transmissão, de acordo com as necessidades do cliente e a situação da rede.

4.1.1 O protocolo "*Loss-delay based adaptation algorithm*"

O protocolo LDA+ (*Loss-delay based adaptation algorithm*) [SISALEM *et al*, 2000a] é um algoritmo AIMD (*Additive Increase and Multiplicative Decrease*) em que os valores de adição e de diminuição da janela de transmissão são determinados pela rede, de forma a adaptar a taxa de transmissão de fluxos, baseado em UDP, em amigável ao TCP.

A magnitude desse aumento, ajustada pelas condições da rede, é orientada pelos seguintes fatores:

- Fluxos com uma pequena largura de banda podem aumentar sua taxa mais rapidamente do que fluxos com largura de banda maior.
- Fluxos devem não exceder o gargalo da largura de banda estimada.
- Fluxos devem não aumentar sua largura de banda mais rapidamente do que uma conexão TCP.

Ao se perceber o congestionamento na rede, o mecanismo de *feedback* passa a ser utilizado para trocar informações sobre perdas, atrasos e a capacidade de banda medida pelo receptor.

Projetado somente para comunicações unicast, o LDA+ usa o protocolo RTP (*Real Time Protocol*) [SCHULZRINNE *et al,* 1996] para trocar informações de *feedback* como, por exemplo: tempo de ida e volta (RTT) e perdas no receptor. O RTP é um protocolo utilizado em transmissões em tempo real, que prove funções de transporte e entrega de dados áudio e vídeo fim-a-fim. Ele é divido em duas partes: uma parte de dados e outra de controle. Na parte de dados, é feita a fragmentação do fluxo de dados, especificando, num cabeçalho adicional, informações de seqüência, tipo de dados e o tempo de entrega. Na parte de controle, ou seja, no RTCP (*Real Time Control Protocol*), é realizado o *feedback* da qualidade de recepção, contendo informações dos dados enviados e recebidos. Essa é uma das características que o RTP proporciona aos protocolos de transporte, como o UDP, necessária para aplicações de mídia contínua. O intervalo de tempo no envio de duas mensagens RTCP é de aproximadamente 5 segundos. Essa freqüência das mensagens de feedback faz com que o remetente RTP não possa se beneficiar rápido o bastante para mudar as condições da rede. Assim, o objetivo do RTCP é ajustar a taxa de transmissão do transmissor para a média da banda disponível e não o de reagir a rápidas mudanças.

Com as informações de *feedback*, o transmissor pode aumentar ou diminuir a taxa de transmissão. Durante os períodos sem perda, o transmissor pode aumentar a taxa de transmissão, através da taxa de aumento aditivo AIR (*Additive Increase Rate*), a qual é estimada usando-se os valores de perda e atrasos, informados pelo *feedback*. O valor próprio do AIR depende do compartilhamento de banda que uma conexão pode utilizar. Inicialmente, é atribuído um pequeno valor que depois é aumentado durante os períodos sem perda. Em tais períodos, o transmissor calcula o AIR_i para esse receptor $_i$ de acordo com a Equação 1:

$$AIR_i = AIR * B_f \text{ , sendo } B_f = 1 - \frac{r}{b} \qquad (1)$$

Onde:

r é a taxa de transmissão atual;

b gargalo da largura de banda;

B_f como sendo o fator da largura de banda, que permite, através da utilização de valores AIR maiores, convergir mais rapidamente para seu estado de compartilhamento justo.

Se alguma perda for notificada pelos receptores, os transmissores colocam o AIR sem seu valor inicial. As perdas detectadas durante a transmissão são calculadas pelo receptor, contando-se os intervalos nos números de seqüência que estão no cabeçalho do RTP. Já o tempo de ida e volta é calculado, conhecendo-se o tempo em que uma informação colocada no transmissor foi gerada e o tempo decorrido entre o recebimento dessa informação e o seu envio pelo receptor. Esses cálculos requerem uma sincronização entre os relógios do transmissor e do receptor.

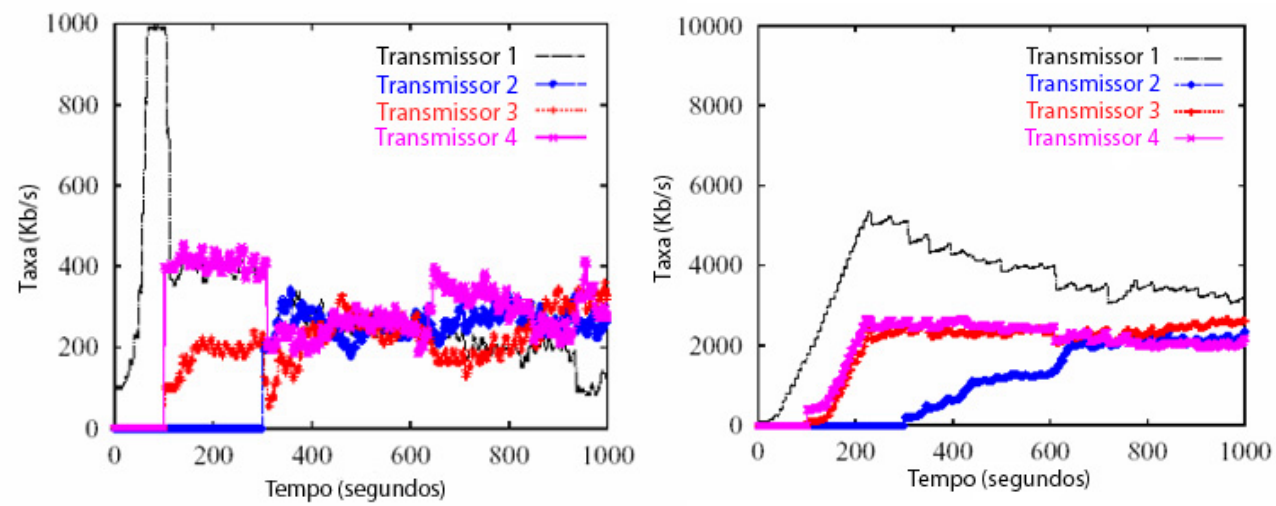
Uma versão mais aprimorada do RTP computa, de acordo com a Equação 2, a largura de banda do gargalo *b,* através de pares de pacotes, em que o transmissor transmite periodicamente uma quantidade de pacotes de dados em rajadas.

$$b = \frac{\text{tamanho do pacote}}{\text{intervalo entre 2 pacotes}} \qquad (2)$$

Resultados de simulações indicam que o uso do protocolo LDA+ [SCHULZRINNE *et al*, 1998] resultou em uma distribuição de banda justa entre as conexões. O experimento realizado consistiu de 4 conexões, compartilhando o mesmo gargalo no roteador, quando os transmissores sempre enviam os dados em uma taxa máxima permitida. O algoritmo implementado no roteador foi o RED (*Random Early Drop*) [JACOBSON *et al,* 1993] que calcula, através do tamanho médio da fila, o início de um congestionamento. Quando esse tamanho excede o valor mínimo fixado, o roteador descarta, com uma certa probabilidade, os pacotes que chegam. Caso o tamanho médio da fila ultrapasse um segundo valor máximo fixado, o roteador, então, descarta toda a chegada de pacotes. O objetivo do uso desse algoritmo foi proporcionar ao teste um limite máximo no tamanho médio da fila e, também, controlar melhor o congestionamento.

Em uma primeira simulação, tendo todas as conexões com o mesmo tempo de ida e volta e inicializadas em tempos diferentes, ficou demonstrado que as 4 conexões recebem uma largura de banda igualmente compartilhada depois de um período de convergência. Com o gargalo na rede configurado para 1 Mbps (figura 2a), o tempo de convergência detectado foi de 300 segundos. Aumentando-se o gargalo do roteador para 10 Mbps (figura 2b) percebeu-se que o tempo de convergência foi em torno de 600 segundos. O tempo de convergência depende do fator

de redução escolhido, o qual determina o grau de reação que o transmissor deve ter, no caso de perdas. Curtos períodos de convergência resultam no uso de valores altos do fator de redução.

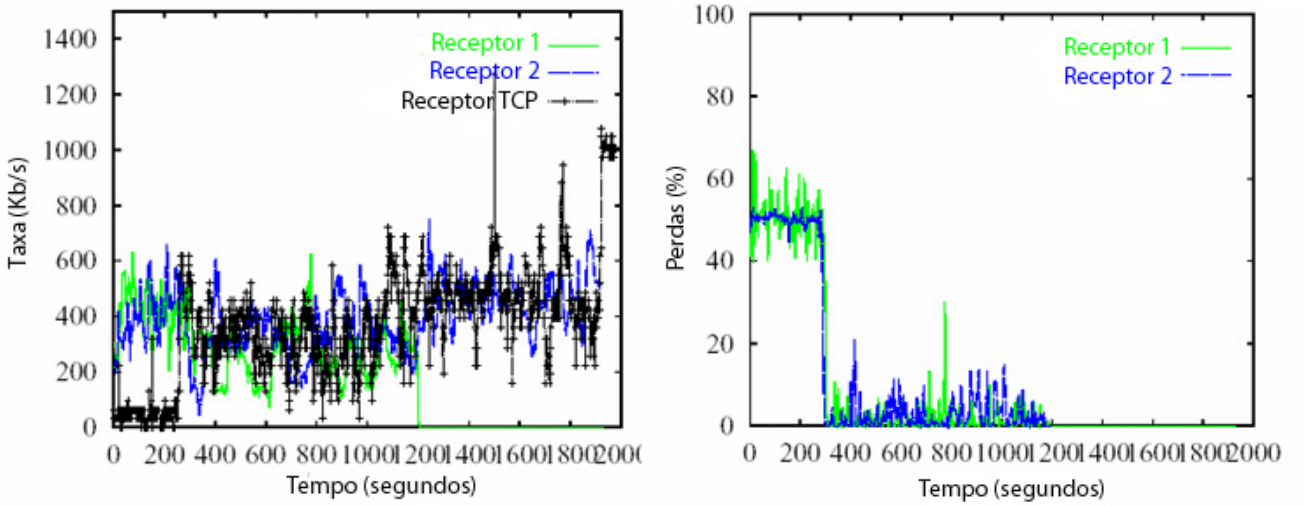


a) b = 1 Mbps b) b = 10 Mbps

Figura 2. Distribuição de banda nas conexões LDA+. [SCHULZRINNE *et al*, 1998]

Em outro experimento de simulação, foi proposta a substituição de um transmissor – que antes eram todos similares – por um transmissor TCP. Com o valor do gargalo no roteador a 10 Mbps e com diferentes RTT, os resultados obtidos foram os de que, no caso do valor do RTT igual a 1 ms, o tráfego TCP recebeu a mesma largura de banda obtida nas conexões LDA+. No caso do RTT igual a 100 ms, percebeu-se que a conexão TCP recebeu a largura de banda perto dos 1.4 Mbps, em que o esperado e o mais justo nesse caso seria 2.5 Mbps. Para a conexão TCP ter alcançado a taxa esperada de 2.5 Mbps seria necessário manter a taxa de perdas em pelo menos 0.16 %. Mas existe um problema, pois o tamanho do campo para o valor da taxa de perda nos pacotes RTCP é de 8 bits, fazendo com que o valor da taxa de perda seja em torno de 0.4 %, resultando na largura de banda perto de 1.6 Mbps. Para que a conexão TCP consiga uma largura de banda justa, exatamente a mesma das conexões LDA+ é preciso uma melhoria no RTCP. Uma das soluções é fazer com que seja usado, ao invés do valor da taxa de perda, apenas um número acumulativo da perda de pacotes, ou seja, o número de pacotes perdidos desde o início do recebimento, permitindo uma indicação mais precisa dessas perdas.

Outros experimentos de simulações, reproduzidos de Schulzrinne [SCHULZRINNE et al, 1998] (figura 3), foram realizados para verificar o desempenho do LDA+ em uma rede local. Dois computadores foram conectados através de um roteador com capacidade limite de 1 Mbps. Foram realizadas duas transmissões de vídeo, com o algoritmo LDA+, de uma máquina para outra e, logo em seguida, foi iniciada uma transferência de arquivos (FTP), em paralelo ao fluxo de vídeo. Nos primeiros 300 segundos, observou-se uma situação típica encontrada na Internet, uma competição entre conexões TCP e UDP, com uma taxa alta de perdas dos transmissores LDA+, em torno de 50 %. Nos 900 segundos seguintes, ambas as conexões LDA+ adaptaram sua largura de banda de uma maneira justa, em torno de 300 Kbps, assim como a conexão TCP. Na conexão TCP, notou-se a queda nas perdas nos transmissores LDA+,chegando agora em torno de 7 %, após 300 segundos do início da transmissão. Quase ao fim da simulação, uma conexão de vídeo foi encerrada, e ambas as conexões agora restantes – UDP e TCP – aumentaram sua largura de banda compartilhada, cada uma com 500 Kbps.



a) Distribuição da largura de banda b) Perdas

Figura 3. Distribuição de banda e suas perdas LDA+. [SCHULZRINNE et al, 1998]

4.1.2 O protocolo "*Rate Adaptation Protocol*"

O RAP (*Rate Adaptation Protocol*) [REJAIE *et al*, 1999] é um protocolo de controle de congestionamento fim-a-fim, baseado em taxa, projetado para entregar qualquer conteúdo de áudio e vídeo armazenados em um servidor Web ou em um servidor de vídeo sob demanda, para diferentes clientes na Internet. A idéia principal do protocolo é separar o controle de congestionamento do controle de erro, pois o controle de congestionamento depende do estado da rede em que se encontra e o controle de erro é especifico da aplicação, de forma que a taxa de transmissão do servidor seja continuamente ajustada pelo protocolo RAP no modo amigável ao TCP.

O protocolo RAP utiliza, também, o método de aumento aditivo e diminuição multiplicativo AIMD para alcançar um estado justo na adaptação da taxa de transmissão. Cada pacote de dados enviado pela origem, onde reside o mecanismo RAP, é reconhecido pelo receptor. Os pacotes de reconhecimento (ACK) contêm o número de seqüência do pacote correspondente entregue e são utilizados para detectar a perda de pacotes e também para coletar amostras do tempo de ida e volta (RTT).

Para projetar mecanismos de adaptação de taxa, três aspectos devem ser considerados: função de decisão, algoritmo de aumento e diminuição e freqüência de decisão.

Função de decisão: Verifica a existência de congestionamento. Realiza, periodicamente, um aumento na taxa de transmissão. Em períodos sem congestionamento a taxa de transmissão é aumentada em 1 pacote por RTT. Caso contrário, ela é diminuída imediatamente, reduzindo pela metade a taxa de transmissão. A busca pela largura de banda disponível acontece de forma semelhante à usada pelo algoritmo de prevenção de congestionamento no TCP (*congestion avoidance*), aumentando linearmente sua taxa de transmissão.

A existência de congestionamento é inferida pela perda de pacotes, o que é detectado pela expiração do intervalo de temporização (*timeout*) e por lacunas na seqüência dos pacotes transmitidos. Esses dois mecanismos trabalham em paralelo e são observados pelo receptor, o qual notifica o transmissor. O transmissor pode enviar vários pacotes antes de receber um novo

ACK para poder atualizar a estimativa do valor de RTT, o qual é chamado de SRTT. Através desse valor atualizado, ele checa a existência de um possível *timeout* entre os pacotes, antes de enviar um novo pacote.

O RAP usa a detecção de perda baseada em tempo para a transmissão de pacotes. A origem mantém um registro para cada pacote transmitido, contendo o número de seqüência, tempo de partida, taxa de transmissão e um *flag* de status. A coleção dos registros dos pacotes que estão sendo transmitidos é chamada de histórico da transmissão.

O mecanismo de detecção de perda baseado em ACK no RAP assemelha-se ao mecanismo *fast recovery* do TCP. Para limitar a quantidade de dados durante a fase de aumento, a origem RAP necessita detectar o congestionamento o mais rápido possível.

No RAP, existe uma maneira para diferenciar a perda de um ACK da perda de um pacote de dados correspondente. Para tal, é adicionada uma redundância nos pacotes de ACK, especificando a última lacuna no espaço de seqüência entregue.

Algoritmo de Aumento e Diminuição: Usando o algoritmo AIMD, a taxa de transmissão é controlada, ajustando-se o *inter-packet-gap* (IPG). Ajustes finos são feitos para deixar o RAP mais estável e responsivo durante um breve congestionamento, baseando-se no cálculo da média do RTT, o que resulta numa taxa de envio mais suave.

O RAP não conta com qualquer sinal explícito de congestionamento vindo da rede. A perda de pacote é o único possível sinal de retorno da rede, devido à presença da competição com o tráfego TCP. Entretanto, se a rede tiver o suporte de sinalização de congestionamento explícito, o protocolo RAP poderá utilizá-lo para tornar o seu comportamento mais eficiente.

Freqüência de decisão: Especifica o quão freqüente a taxa de transmissão é alterada. Encontrar o melhor ajuste na freqüência da alteração da taxa depende do retorno do atraso que é recebido. Nos modelos baseados em taxa como o RAP, os ajustes na taxa não devem ser maiores que 1 por RTT, pois ajustes muitos freqüentes podem causar oscilações na rede.

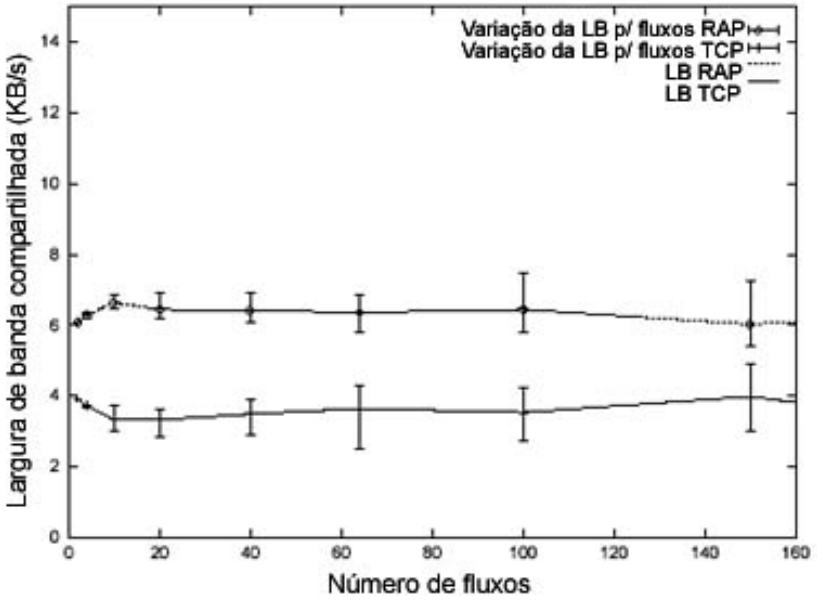
Para o transmissor inferir como a rede se comportará com um novo ajuste, antes mesmo de ele ser feito, o valor do IPG é ajustado uma vez a cada RTT, variando de acordo com o estado

de congestionamento. Caso nenhuma perda seja detectada, o IPG é diminuído e o tempo entre dois pontos de ajustes subseqüentes, chamado de *step*, é iniciado.

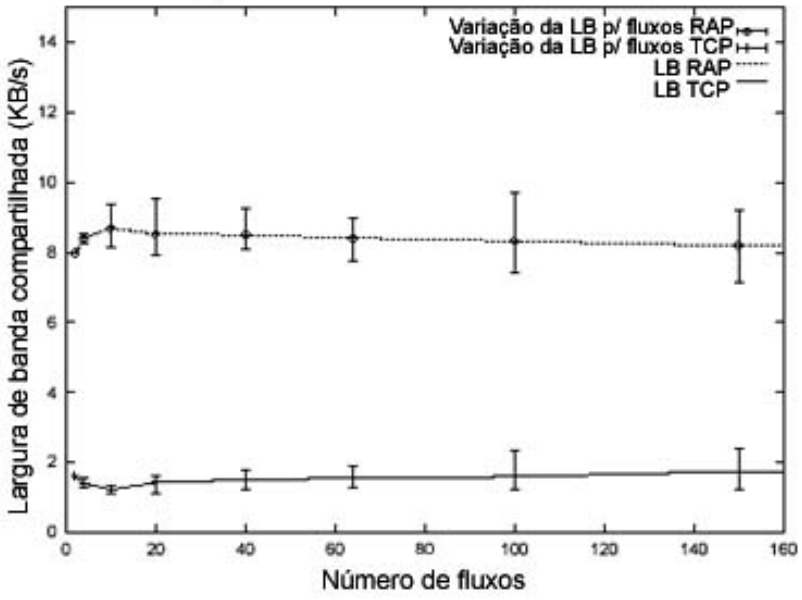
Resultados da simulação feitos com o RAP mostraram que este protocolo é amigável ao TCP em uma grande variedade de cenários, possuindo habilidade de cooperar com tráfegos TCP e interagir com roteadores usando o algoritmo RED. Todos os fluxos TCP utilizados foram sessões de transferências de arquivos com uma quantidade contínua de dados. Em um ambiente com uma grande quantidade de parâmetros é geralmente difícil isolar uma variável em particular para estudar e distinguir principalmente os efeitos causados pelos fluxos TCP, quando coexistindo com fluxos RAP. Para minimizar o problema, considerou-se nos experimentos a comparação do RAP com várias implementações do TCP.

Observou-se nos experimentos realizados com os fluxos RAP e as implementações do TCP Tahoe, Reno, NewReno e Sack, um certo grau de injustiça contra os tráfegos TCP, devido a limitações de desempenho herdadas pelo TCP. Essas limitações acontecem quando há muitas perdas dentro de uma janela, o que força o TCP a esperar pelo *timeout* de retransmissão ou a iniciar diretamente o algoritmo *slow-start*.

Dentre as implementações de TCP utilizadas nas simulações (figura 4), a implementação Sack foi a que obteve um melhor desempenho, pois ela consegue tratar com mais eficiência os muitos segmentos perdidos de uma janela. A média da largura de banda compartilhada nos protocolos RAP e TCP, variando-se o número total de fluxos, foi de 6 Kbps e 4 Kbps respectivamente. A implementação Reno foi a que obteve o pior resultado, com média de 8 Kbps para fluxos RAP e 2 Kbps para fluxos TCP, independente da variação do número de fluxos.



a- RAP coexistindo com o Sack



b- RAP coexistindo com o Reno

Figura 4. Comparação do desempenho com as implementações Sack e Reno. [REJAIE *et al,* 1999]

Dois parâmetros essenciais para o comportamento dos tráfegos são as variações nos valores de RTT e a janela de congestionamento. Valores muitos altos de RTT podem causar retransmissões desnecessárias que afetam o ajuste da taxa no RAP e a performance do protocolo TCP. s simulações compreenderam a relação entre os cálculos das larguras de banda de todos os fluxos RAP sobre os fluxos TCP em função do atraso e do número de fluxos. Vários atrasos no gargalo do enlace foram realizados e revelaram que o aumento no número de fluxos (> 120 fluxos) melhora a razão de justiça, convergindo para o valor 1. O fato de alcançar essa razão de justiça deve-se ao grande número de pacotes em trânsito, o que permite ao TCP recuperar-se facilmente de múltiplas perdas nas janelas de congestionamento. Notou-se, de acordo com a reprodução da simulação realizada por Rejaie [REJAIE *et al,* 1999] (figura 5) que, para atrasos pequenos (20 ms), o número de fluxos não surtiu efeito e a razão de justiça ficou entre 1,5 e 2.

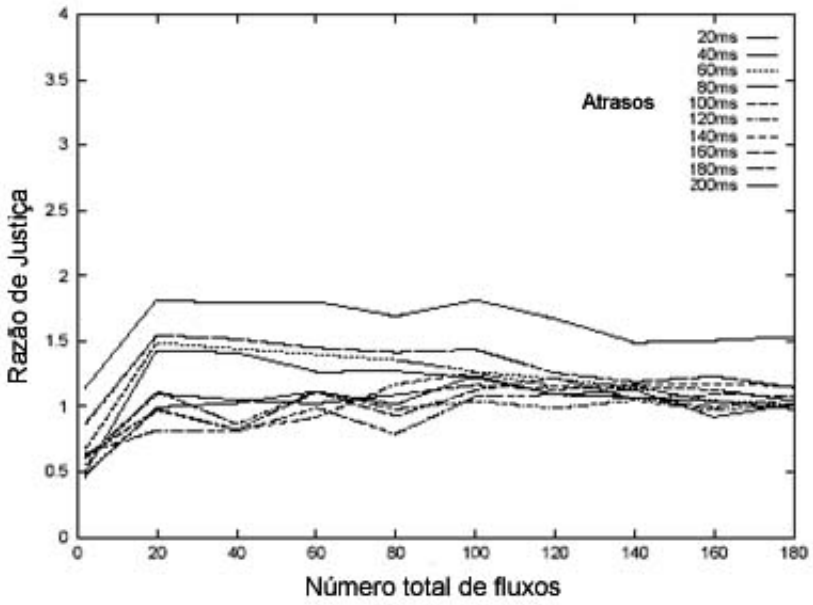


Figura 5. Grau de Justiça em relação ao número de fluxos RAP. [REJAIE *et al,* 1999]

Através do uso de um ajuste fino na taxa, característica adicional do RAP, foi possível obter um grau de justiça, também convergindo para o valor 1, das conexões com o atraso pequeno (20 ms). Diante desse resultado, tem-se que a razão de justiça é determinada, primeiramente, pelo comportamento do TCP e, em casos em que o TCP é vulnerável a múltiplas perdas, o ajusto fino no RAP previne o uso de toda banda compartilhada disponível, conseguindo o TCP obter um compartilhamento justo de tráfego.

4.1.3 O protocolo "TCP *emulation at receivers*"

O TEAR (TCP *emulation at receivers*) [RHEE *et al*, 2000] é um protocolo de controle de fluxo fim-a-fim que implementa a técnica de controle de fluxo no receptor, emulando as funções de controle de fluxo do TCP, como por exemplo: *slow start, fast recovery e congestion avoidance*.

No protocolo TEAR, o receptor não envia nenhuma notificação de detecção de congestionamento ao emissor. A notificação de congestionamento é processada no receptor, o mais rápido possível, determinando qual será sua própria taxa de recebimento. Essa taxa calculada é utilizada no receptor para fazer o controle de congestionamento, sem receber nenhuma informação de *feedback*. Esse cálculo da taxa baseia-se em inferências do congestionamento observado no receptor como, por exemplo, a chegada de pacotes, perdas de pacotes e *timeouts*.

Emulando no receptor as funções do controle de fluxo do TCP, o protocolo TEAR ajusta a taxa de recebimento para uma taxa amigável ao TCP, sem sondar antes por uma largura de banda livre ou reagir diretamente a perdas de pacotes. Desta forma, a oscilação na taxa de transmissão permanece mais suave do que no protocolo TCP.

No TEAR, chama-se de independência à taxa a probabilidade de que um pacote seja perdido dentro de uma janela de x pacotes transmitidos consecutivamente. Essa independência à taxa não depende da sua taxa de transmissão, isto é, não importa o quão grande ou pequeno seja o intervalo em que eles são transmitidos, a probabilidade de que pelo menos um pacote na janela seja perdido é o mesmo, desde que as condições da rede não sejam alteradas durante todo o período de transmissão. A definição independência de taxa supõe que as perdas de pacotes aconteçam independentemente, pois nos roteadores eles são descartados de maneira indiscriminada.

Os pacotes no protocolo TEAR são transmitidos a uma taxa diferente de uma conexão TCP. Pode-se avaliar uma conexão TCP sobre um mesmo caminho fim-a-fim somente pela

observação dos pacotes que chegam ao receptor. Isso implica que uma janela de *n* pacotes em uma conexão TCP tem a mesma probabilidade de perda que uma conexão feita pelo protocolo TEAR, não levando em conta a taxa de transmissão.

O TEAR mantém uma variável chamada *cwnd*, indicando o número de pacotes em trânsito, e que é atualizada a cada pacote que chega ao receptor. As atualizações das janelas de congestionamento, nesse caso, não podem ser descritas em termos de tempo de ida e volta (RTT) e sim, no tempo de transmissão em uma única direção, pois tanto o TEAR como o TCP podem transmitir em diferentes taxas.

No TEAR, o receptor pode reconhecer a transmissão em uma única direção (*round*) quando o pacote é recebido, enquanto no TCP ele é reconhecido no transmissor, quando um pacote de reconhecimento dos pacotes de dentro de uma janela de congestionamento (*cwnd*) é recebido. No final de cada *round*, o receptor grava o valor corrente do *cwnd* em uma estrutura de dados que será utilizado para calcular a taxa amigável ao TCP, ilustrado na figura 6.

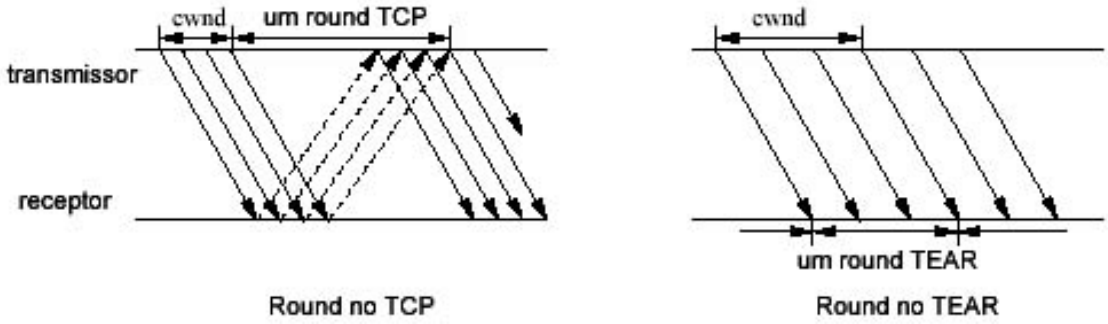


Figura 6. *Round* no TEAR e no TCP. [RHEE *et al*, 2000]

Algumas funções do controle de fluxo do TCP são emuladas pelo TEAR. No caso do algoritmo de *slow start,* o valor da variável *cwnd* é duplicado a cada *round*. Já quando o estado dos pacotes entra em *congestion avoidance,* o valor da variável *cwnd* e incrementado de 1 a cada *round*.

A figura 7 compara o comportamento do *fast recovery* no TCP e no TEAR. No TCP, a fase de *fast recovery* é alcançada quando três ou mais ACKs duplicados são recebidos após uma perda de um pacote. No TEAR, a fase de *fast recovery* ocorre quando, pelo menos dois pacotes

são recebidos, antes da chegada de um pacote com número de seqüência maior que a soma do número de seqüência do último pacote recebido mais o valor do último *cwnd*.

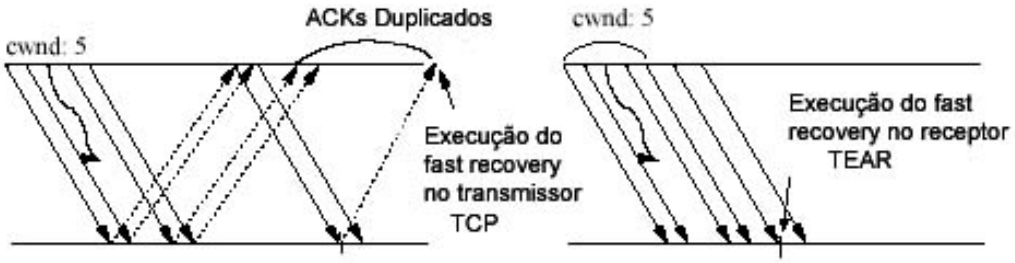


Figura 7. Comparação da fase Fast Recovery no TCP e TEAR [RHEE *et al*, 2000]

Para evitar o comportamento dente-de-serra, causado pelo aumento aditivo e diminuição multiplicativa (AIMD), o TEAR calcula a média da taxa sobre uma época. Uma época é definida pelo período entre os eventos de redução de taxa consecutivos, e este não deve ser pequeno demais para que a taxa não oscile muito e nem grande demais para que o ajuste da taxa não seja percebido pela rede.

Os experimentos realizados contaram com vários valores de parâmetros como largura de banda, números de fluxos TCP, números de fluxos TEAR e RTT. Foram analisados o grau de justiça da coexistência de fluxos TCP e TEAR. Notou-se um melhor desempenho, utilizando-se o algoritmo RED no roteador, em relação a um outro algoritmo chamado *Drop Tail*. Esse *Drop Tail* funciona de forma a descartar os pacotes no fim da fila que, estando cheia, descarta os outros que vão chegando.

A simulação realizada com uma largura de banda do gargalo de 2.5 Mbps utilizou o algoritmo *Drop Tail* nos roteadores. Foram inicializadas 16 transmissões com o protocolo TCP e 1 com o protocolo TEAR. Os resultados mostraram que a transmissão TEAR usou uma taxa abaixo (aproximadamente 10 Kbps) do nível de justiça compartilhado, obtida pela simulação em torno de 19 Kbps, conforme a figura 8.

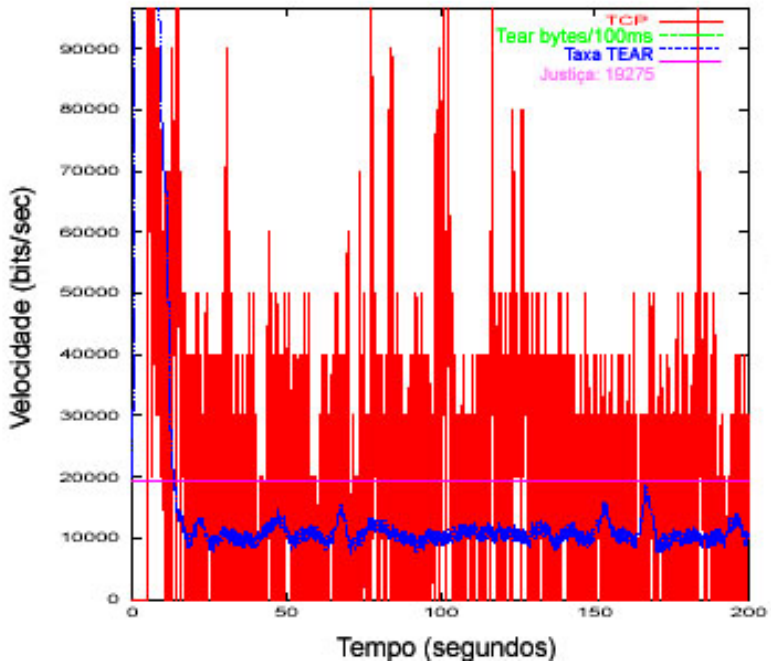


Figura 8. Fluxos TEAR e TCP utilizando a implementação Drop Tail. [RHEE *et al*, 2000]

Repetindo a simulação com o mesmo número de transmissões de ambos os protocolos, utilizando agora o algoritmo RED, notou-se que a taxa do TEAR seguiu suficientemente bem a linha da taxa de justiça, ou seja, todas as transmissões seguiram a taxa em torno de 19 Kbps (figura 9).

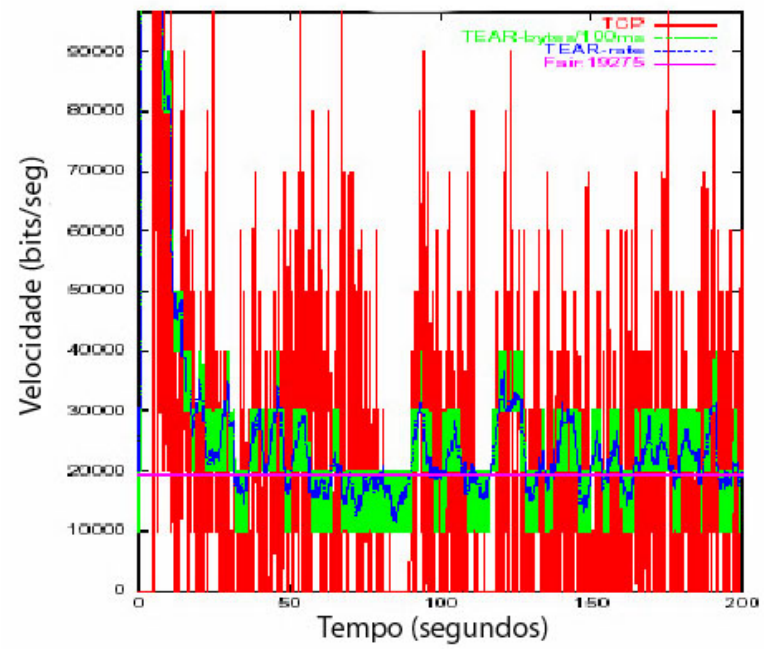


Figura 9. Fluxos TEAR e TCP utilizando a implementação RED. [RHEE *et al*, 2000]

O TEAR comportou-se muito justo em todas as simulações feitas com uma largura de banda do gargalo maior (10 Mbps). Com relação a oscilações na taxa, as simulações mostraram poucas oscilações no fluxo TEAR, comportando-se muito estável.

4.1.4 O protocolo "TCP-*Friendly Rate Control*"

O protocolo *TCP Friendly Rate Control*, TFRC, [HANDLEY *et al*, 2003] é um protocolo de controle de congestionamento para tráfegos *unicast*, para ser utilizado em conjunto com um protocolo de transporte. Foi desenvolvido para trabalhar de forma justa e competitiva com fluxos TCP, operando em ambientes de melhor esforço, como a Internet.

Esse mecanismo de controle de congestionamento não tem por objetivo utilizar toda a banda disponível para transmissão, mas o de manter relativamente estável a taxa de transmissão, enquanto é elaborada uma resposta ao congestionamento detectado na rede. Ao contrário do TCP, sua taxa de transmissão não deve ser dividida ao meio, em resposta a uma sinalização de uma simples perda de pacote; isso acontecerá somente quando houver várias perdas sucessivas durante a transmissão.

O TFRC é implementado no receptor que registra todas as características importantes da rede, as quais serão utilizadas para determinar se a taxa de envio deve ser aumentada ou diminuída. Todos esses parâmetros são mantidos apenas no receptor, não sendo necessário seu conhecimento pelo transmissor.

A obtenção do valor da taxa de um evento de perda é de extrema importância para o TFRC. Um evento de perda acontece quando não existe uma largura de banda suficiente entre o transmissor e o receptor, e pacotes devem ser descartados, sendo composto por um ou mais pacotes perdidos durante um RTT. A taxa é calculada no receptor, utilizando-se a detecção de perda da seqüência dos pacotes que chegam até o receptor. Todos os pacotes de dados possuem um número de seqüência, que é acrescido de um, para cada pacote de dados transmitido, além da informação de RTT atual do transmissor e também um *timestamp* indicando quando o pacote foi enviado.

Quanto mais cedo for detectado um evento de perda, tanto mais cedo será garantida uma resposta ao congestionamento, portanto o protocolo espera por três ou mais pacotes depois de receber um pacote fora do número de seqüência. Se todos os três números de seqüência forem maiores do que o número de seqüência que está faltando, esse pacote ausente é considerado perdido e o valor da taxa de evento de perda é atualizado.

Os métodos adotados para realizar as medidas das perdas são:

- ***Janela dinâmica de perdas*:** um método básico para determinar a taxa de evento de perda é medir o número de eventos de perdas sobre um número fixo de pacotes, utilizando a técnica de janela deslizante. O tamanho da janela deve ser ajustado dinamicamente para as condições atuais, para que um valor alto no tamanho da janela não faça com que o protocolo TFRC se adapte muito lentamente às mudanças feitas na taxa de eventos de perdas. Esse método possui desvantagens e requer um esforço muito grande na sua implementação.

- ***Média exponencial:*** um método simples e justo usado para calcular as perdas, utilizando a média exponencial do número de pacotes entre os eventos de perdas.

- ***Intervalo de média ponderada de perdas:*** um método alternativo ao da média exponencial, utilizando a média aritmética dos números de pacotes de *n* intervalos de perda, ao contrário do número de eventos de perdas. Por obter um resultado mais preciso, ele é utilizado atualmente no protocolo TFRC.

O aumento na taxa de transmissão deve ocorrer lentamente em resposta a uma diminuição na taxa de perda. Essa suave variação na taxa em comparação com o TCP faz o protocolo TFRC ser muito mais apropriado para aplicações de fluxos vídeo e áudio que utilizam um tamanho de pacote fixo, em que a estabilidade da taxa de envio é relativamente importante.

Utilizando a Equação 3 [PADHYE *et al*, 2000], o TFRC ajusta a taxa de envio no transmissor em função da taxa de perda, RTT e o tamanho do pacote, resultado numa taxa de transmissão aceitável.

$$T = \frac{s}{R\sqrt{\frac{2bp}{3}} + (t\_RTO\ (3\sqrt{\frac{3bp}{8}})\ p\ (1 + 32p^2))} \qquad (3)$$

Onde:

T: taxa de transmissão em bytes por segundo.

s: tamanho do pacote em bytes.

R: round trip time (RTT).

p: taxa do evento de perda.

t_RTO: valor do *timeout* de retransmissão TCP em segundos.

b: número de pacotes reconhecidos.

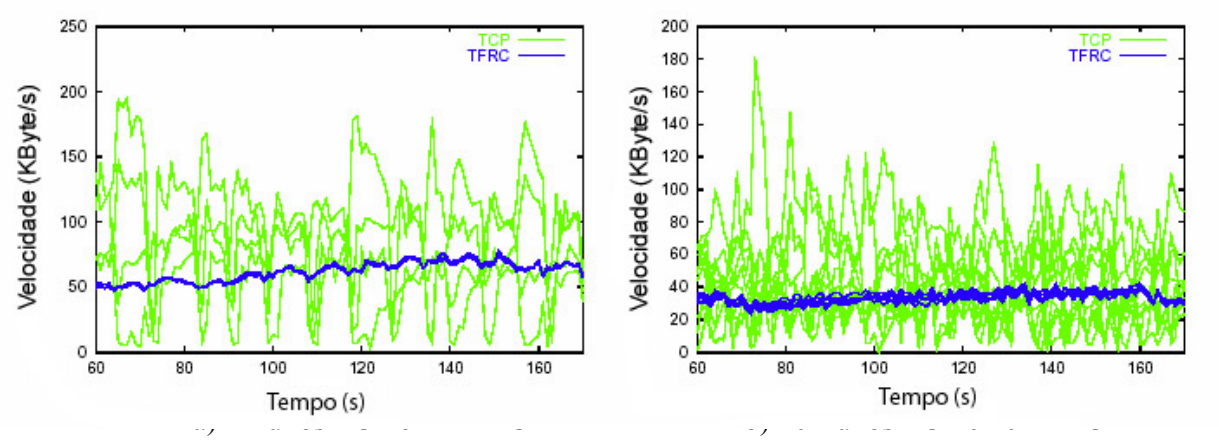
O receptor deve enviar, pelo menos uma vez a cada RTT, um pacote de *feedback,* indicando se os pacotes foram recebidos naquele intervalo. Caso o transmissor não receba pacotes de *feedback* após vários RTTs, o transmissor deverá reduzir sua taxa de transmissão.

Cada pacote de *feedback*, enviado pelo receptor, contém informações como: quantidade de tempo decorrido entre o recebimento do último pacote de dados e a geração do *feedback*, o *timestamp* do último pacote de dados recebido e a estimativa atual da taxa de perda. Os pacotes de *feedback* também são aproveitados pelo transmissor para medir o seu RTT através da rede.

Simulações foram realizadas para saber como o protocolo TFRC compete e interage com fluxos TCP. Um parâmetro importante para a análise do comportamento do protocolo é a garantia de que o tamanho dos pacotes do TFRC seja igual aos do TCP, pois estão intimamente

relacionados. As simulações reportadas por Widner [WIDNER 2000] foram feitas através da Internet, com um link T1 de 1,5 Mbps, entre hosts das Universidades de Massachusetts (UMASS), Mannheim (UMANN) e College London (UCL), situadas a uma grande distância, com uma combinação de diferentes quantidades de fluxo.

Na simulação feita na UCL (figura10), foram combinadas diferentes quantidades de fluxos TCP e TFRC. Na simulação realizada com 4 fluxos TCP e 4 fluxos TFRC e, posteriormente, ambos com 10 fluxos, o TFRC comportou-se próximo ao nível de justiça com o compartilhamento de 44% de toda largura de banda em experimentos com 4 fluxos e 45 % em experimentos com 10 fluxos. Nota-se que o TFRC teve um comportamento parecido em ambas as quantidades de fluxos.

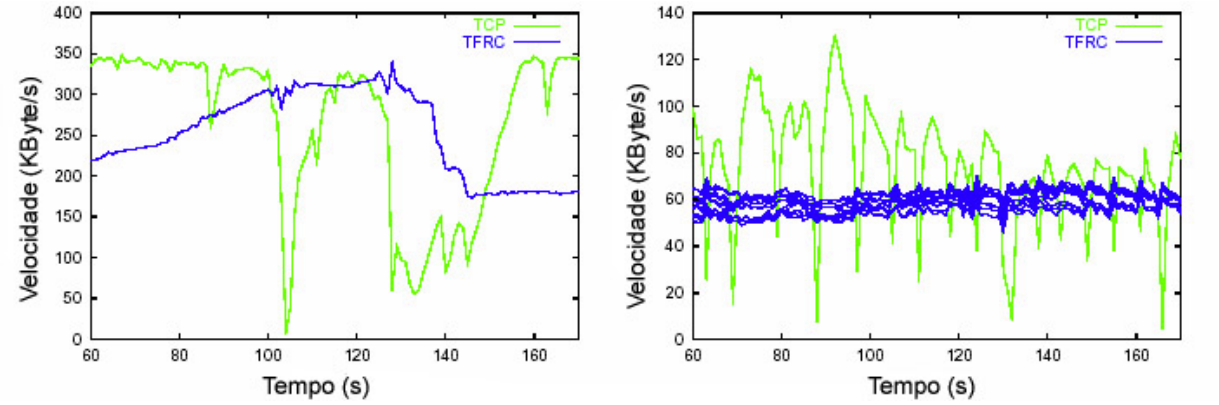


a- 4 fluxos TCP e 4 TFRC

b- 10 fluxos TCP e 10 TFRC

Figura 10. Diversos fluxos TCP e TFRC compartilhando a mesma banda. [WIDNER 2000]

Experimentos realizados na UMANN (figura 11), com 1 fluxo apenas em ambos os protocolos TFRC e TCP, realçaram a diferença entre os dois protocolos. Após 100 segundos do inicio da simulação, notou-se um congestionamento de curta duração na rede, o qual gerou um *timeout* na conexão TCP. O fluxo TFRC continuou a se ajustar normalmente. Um outro congestionamento, agora de longa duração, ocorreu aos 125 segundos do início da simulação e forçou a reação do protocolo TFRC, reduzindo suavemente sua taxa de 300 Kbyte/s para 160 Kbyte/s. Ao cessar o congestionamento, percebeu-se que o TCP voltou à sua taxa original mais rapidamente que o TFRC, pois o TFRC necessita de um pouco mais de tempo, até que seja feito o ajuste do evento de perda.



a) 1 fluxo TCP e 1 TFRC b) 1 fluxo TCP e 10 fluxos TFRC

Figura 11. Comparação do TFRC com apenas 1 fluxo TCP. [WIDNER 2000]

Na simulação realizada na UMASS (figura 12), notou-se que, durante o experimento compartilhando-se 4 fluxos TCP e 4 fluxos TFRC, houve um aumento momentâneo na largura de banda compartilhada disponível de 30 Kbyte/s para 100 Kbyte/s, após 120 segundos do início da simulação. Esses aumentos na largura de banda compartilhados foram acompanhados, simultaneamente, pelos protocolos TCP e TFRC, comportando-se ambos de maneira semelhante durante a simulação.

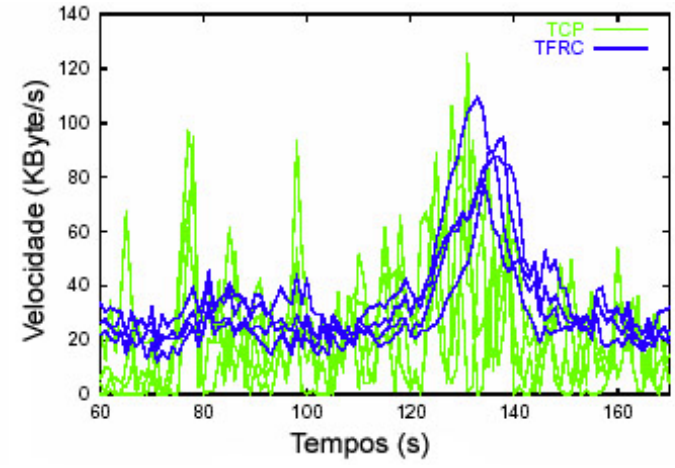


Figura 12. Fluxos TCP e TFRC sob um aumento na largura de banda. [WIDNER, 2000]

4.2 Protocolos *Multicast* Amigáveis ao TCP

Os protocolos *multicast* amigáveis ao TCP são utilizados em aplicações em que um único transmissor deve servir o mesmo fluxo a um grande número de receptores. O desenvolvimento de controle de congestionamento para transmissões *multicast* é muito mais trabalhoso do que para transmissões *unicast*, pois envolve um grande número de receptores, e ainda deve ser capaz de operar em redes heterogêneas.

4.2.1 O protocolo "*Random Listening Algorithm*"

O protocolo RLA (*Random Listening Algorithm*) [WANG *et al*, 1998] é um mecanismo de controle de congestionamento *multicast,* baseado em janela e projetado para funcionar tanto em roteadores com implantação RED quanto em roteadores *Drop Tail*. Possui grande semelhança com o TCP e utiliza procedimentos de reconhecimento seletivo (SACK). O RLA procura obedecer aos requisitos mínimos para manter um compartilhamento de banda justo, de forma a não encerrar fluxos TCP ou diminuir a produtividade de uma sessão *multicast* para zero, quando o número de receptores aumenta.

Para alcançar esses objetivos, é preciso localizar em qual caminho da árvore se localiza o gargalo de uma sessão *multicast,* para que se possa reagir a uma notificação de congestionamento. No entanto, é difícil realizar essa localização, baseando-se somente na informação de perdas, devido ao fato de que tanto o transmissor quanto o receptor calculam a provável perda para cada receptor. Uma reação terá que ser elaborada somente para aquela perda que foi informada. Uma ação tomada em um caminho que não apresenta gargalo pode causar resultados indesejáveis. Foi desenvolvido, então, um raciocínio que consiste na troca da busca de um caminho congestionado por uma rápida resposta ao sinal de congestionamento.

Cada receptor que apresenta uma situação de congestionamento envia ao transmissor uma notificação desse congestionamento. A redução no tamanho da janela de congestionamento, a cada notificação, obviamente diminuirá a produtividade de uma sessão *multicast,* de acordo com aumento do número de receptores. Uma solução viável é todas as conexões começarem simultaneamente. Assim, o transmissor recebe todas as notificações de congestionamento de uma

maneira sincronizada, sejam quantos forem os receptores, reduzindo uma única vez a janela de congestionamento a cada período. No entanto, os reconhecimentos de congestionamento chegam em tempos diferentes, de forma assíncrona, em uma seqüência muito irregular motivo pelo qual essa solução não produziu efeitos positivos.

A proposta de um algoritmo de monitoramento aleatório para tratar o fluxo de notificações de congestionamento é utilizada no protocolo RLA, para produzir um bom desempenho no compartilhamento de banda. Ao receber uma notificação de congestionamento, o transmissor reduz sua janela com probabilidade $1/n$, onde n é o número de receptores que informam sobre suas freqüentes perdas. Se o transmissor detectar um receptor em situação de congestionamento intensa e os outros em melhor condição , a janela de congestionamento é reduzida menos freqüentemente, comparada com o TCP, resultando numa maior proporção no tamanho da janela e limitando o compartilhamento de banda *multicast* em torno do compartilhamento de banda TCP.

A estrutura do algoritmo RLA é a seguinte:

- **Detecção de perda:** os receptores utilizam o mesmo formato de reconhecimento seletivo (SACK) realizado no TCP. Uma perda é detectada pelo transmissor, ao receber uma descontinuidade no número de seqüência ACK ou um *timeout*.
- **Detecção de congestionamento:** quando uma perda é detectada, inicia-se um período no qual os pacotes são descartados. Perdas com duas vezes o valor de RTT são agrupadas em um reconhecimento de congestionamento.
- **Ajuste na janela:** ao detectar o congestionamento, a contagem dinâmica do número de receptores que estão informando perdas é atualizada. Se a probabilidade de perdas de um receptor for pequena ele não é computado como um receptor com problemas. Sempre que um pacote é reconhecido por todos os receptores, a janela de congestionamento (*cwnd*) é ajustada para: $cwnd \leftarrow cwnd + 1/cwnd$.

Para evitar um longo crescimento da janela de congestionamento, o qual não é desejável, o corte forçado da janela ($cwnd \leftarrow cwnd/2$) entra em ação, sempre quando o tempo reduzido à

pela última vez for além do valor de 2 x *awnd* x $srtt_i$ (onde *awnd* é a mudança média do tamanho da janela e $srtt_i$ é o valor do RTT entre o transmissor e o receptor *i)*.

Uma topologia de rede em árvore de quatro níveis foi criada por Wang [WANG *et al*, 1998] para simular o comportamento do protocolo RLA quando em competição com fluxos TCP. As simulações foram implementadas através do simulador de rede NS2 (*Network Simulator*), que é um software para simulações de protocolos e análises de comportamento de redes. Os transmissores, tanto do RLA como do TCP, foram colocados na raiz da árvore (nó S) e os receptores numerados de R1 até R27. Os nós entre os níveis são roteadores (Gx) que podem ser tanto do tipo RED ou *Drop Tail*, conforme ilustra a figura 13.

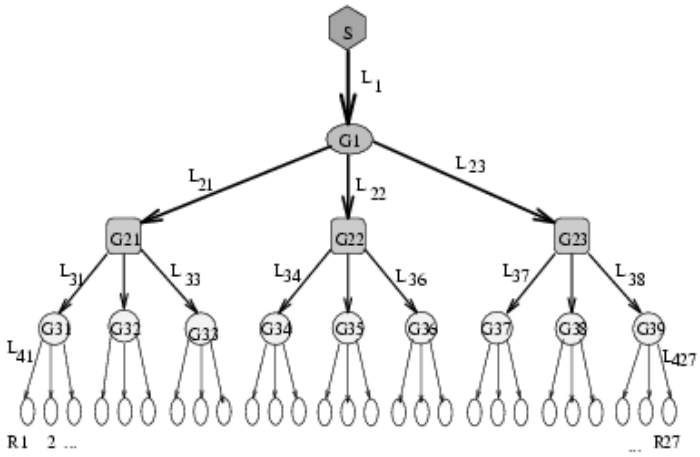


Figura 13. Topologia RLA em árvore de 4 níveis. [WANG *et al*, 1998]

Os resultados da simulação feita em roteadores, utilizando a implementação RED apresentaram os melhores resultados de um justo compartilhamento entre o TCP e a sessão *multicast.* Foram criadas 5 situações de gargalo nos enlaces (Lx) em diferentes níveis e os resultados do protocolo RLA foram comparados com o melhor resultado TCP obtido. A Tabela 1 foi gerada contendo os resultados de velocidade em pacotes por segundos, tamanho da janela de congestionamento, RTT, número de notificações de congestionamento e número de cortes forçados na janela.

Situações		1	2	3	4	5
Enlaces Congestionados		L1	L3i, i=1,.....,9	L4i, i=1,.....,27	L4i, i=1,.....,5	L21
R L A	Velocidade (pacotes/segundos)	118,0	103,7	88,3	141,0	209,2
	Tamanho da janela de congestionamento	27,6	27,0	25,9	36,3	49,6
	RTT (segundos)	0,233	0,264	0,283	0,261	0,236
	No. de notificações de congestionamento	25272	19188	19895	13939	12132
	No. corte forçado na janela.	949	729	721	545	454
T C P	Velocidade (pacotes/segundos)	88,9	86,1	74,0	166,2	576,4
	Tamanho da janela de congestionamento	21,5	22,6	21,1	41,8	135,7
	RTT (segundos)	0,232	0,249	0,265	0,243	0,231
	No. corte forçado na janela.	812	707	702	433	178

Tabela 1. Tabela com os resultados da simulação com a implementação RED.

Na comparação com as situações 1,2 e 3 observa-se uma correlação muito próxima entre tamanho da janela de congestionamento e velocidade. Altos valores no tamanho da janela resultaram em altas velocidades em ambos os protocolos.

Nas situações 4 e 5, notou-se uma grande discrepância na freqüência de congestionamento entre o RLA e o TCP, devido ao tamanho das janelas de congestionamento ser muito diferente. Em todas as situações, um grande valor no tamanho da janela de congestionamento proporciona uma maior probabilidade na redução da janela na metade.

4.2.2 O protocolo "*Multicast* TCP"

O protocolo *Multicast* TCP, MTCP, [RHEE *et al*, 1999] é um protocolo de controle de congestionamento baseado em janela. Foi desenvolvido para tratar de maneira confiável a transmissão dos dados a um grande número de pessoas participantes (receptores) envolvidos, quando se utiliza a tecnologia *Multicast*. Foi projetado para manter a compatibilidade com fluxos

TCP, visto ser o protocolo de transporte mais comum na Internet, e também por ser escalonada frente ao grande número de aplicações envolvidas pelos diversos receptores.

O protocolo MTCP possui uma estrutura de árvore lógica, cuja raiz tem o papel de transmissor e os outros nós da árvore são os receptores. Os nós internos da árvore são chamados de agentes transmissores (*Sender's Agents*) que têm a responsabilidade de lidar com os pacotes de *feedback* gerados por seus filhos e retransmitir os pacotes perdidos.

Cada receptor pode enviar um reconhecimento positivo (ACK) ou negativo (NACK). Um agente transmissor, ao receber de seus nós filhos um ACK, descarta todos os pacotes que estão em seus *buffers,* os quais foram recebidos do transmissor. Caso contrário, ou seja, ao se receber um NACK, a falta de um pacote ou o não-recebimento de um ACK de todos filhos de uma árvore, antes da expiração do tempo associado a cada pacote, força-se o agente transmissor a retransmitir rapidamente o pacote via *unicast,* diretamente ao receptor que reportou o NACK. Para indicar com exatidão qual pacote chegou ao receptor, o protocolo MTCP utiliza o esquema de reconhecimento seletivo (SACK), prevenindo o reenvio desnecessário de pacotes, enviando somente aquele perdido pelos seus filhos.

Para garantir que uma sessão MTCP não apresente um congestionamento em qualquer dos caminhos, é importante que o transmissor regule a taxa de transmissão baseado em um sumário de congestionamento que é carregado por um pacote de reconhecimento ao longo do caminho, dos filhos até a raiz da árvore lógica, obedecendo a um esquema hierárquico. Caso algum enlace da árvore de roteamento *multicast* esteja congestionado, o MTCP diminuirá sua taxa de transmissão.

Um pacote de reconhecimento, enviado pelo agente transmissor ao seu nó pai, carrega um sumário do nível de congestionamento dos seus filhos. Esse sumário inclui uma estimativa da largura de banda mínima disponível ao longo do caminho do transmissor até os seus receptores e que consiste no tamanho da janela de congestionamento *(cwnd)* e do número de bytes em trânsito do transmissor até os filhos do agente transmissor *(twnd)*.

O tamanho da janela de congestionamento comporta-se de forma semelhante ao do TCP, incluindo o controle de congestionamento *slow start* e *congestion avoidance*. No entanto, existem duas diferenças em relação ao TCP:

- A janela de congestionamento é aumentada quando o agente transmissor recebe os ACKs de todos os seus filhos.
- Quando um receptor envia um NACK relatando a perda de um pacote, o agente transmissor retransmite imediatamente o pacote perdido.

Já o número de *bytes* em trânsito é incrementado, sempre que um novo pacote é recebido do transmissor, e diminuído quando um pacote é reconhecido por todos os filhos de um agente transmissor.

O sumário de congestionamento de um agente transmissor consiste do seu próprio valor da janela de congestionamento, do valor mínimo recebido pelos seus filhos (*mincwnd*) e do máximo valor de número de *bytes* em trânsito dos seus filhos (*maxtwnd*). Através deste mecanismo de janelas, o transmissor envia sempre os dados a serem transmitidos numa quantidade menor que a diferença dos valores *maxtwnd* e *mincwnd*, prevenindo que se transmita o pacote de dados mais rápido que o gargalho do enlace possa suportar.

Se muitos pacotes forem perdidos e retransmissões se tornarem necessárias, ocorrerá congestionamento na rede. Devido à possibilidade de os novos pacotes retransmitidos poderem escolher rotas diversas, a janela de congestionamento não pode ser utilizada para controlar o tráfego desses pacotes. Para solucionar esse problema no MTCP, cada agente transmissor mantém uma outra janela chamada de janela de retransmissão, destinada a cada filho, usada somente para os pacotes retransmitidos que são atualizados da mesma forma das janelas de congestionamento.

No MTCP, ao contrário do TCP, não se pode medir o RTT de um pacote devido a variações na rede e ao fato de que, tanto o relógio do transmissor quanto os dos agentes transmissores não estarem sincronizados. Para estimar o valor do intervalo de temporização de retransmissão dos pacotes, foi introduzido o conceito de *Relative Time Delay* (RTD) que é a diferença entre o valor do relógio do transmissor quando um pacote é enviado e o valor do

relógio de um agente transmissor, quando o correspondente ACK é recebido de um nó filho, tendo, assim, a cada recebimento de um ACK, um valor de RTD.

Experimentos foram realizados para demonstrar como o protocolo MTCP se comporta, interagindo com tráfegos TCP. Foram criados cinco grupos, conforme mostra a árvore *multicast* na figura 15, em diferentes localidades, e cujos pacotes, gerados pelo transmissor na raiz da árvore, foram encaminhados para os seus filhos via UDP.

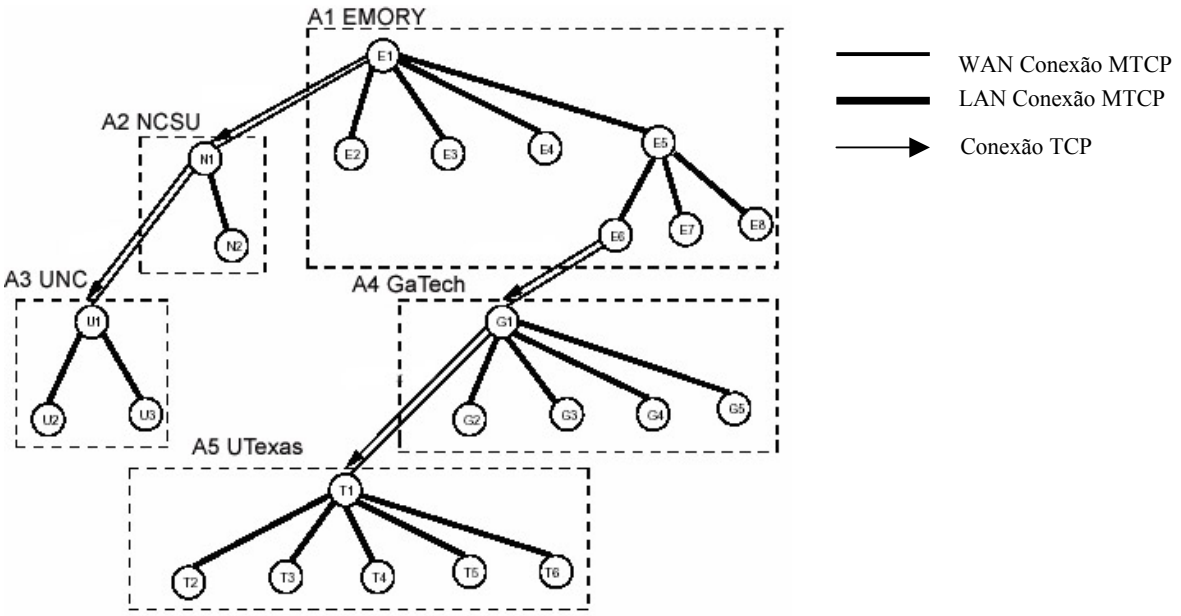


Figura 14. Árvore *multicast* MTCP dividido em 5 áreas. [ROUSKAS *et al*, 1999]

Os enlaces de longa distância (WAN) que interconectam as localidades foram o alvo das simulações, pois é onde os tráfegos MTCP e TCP compartilham o mesmo caminho, servindo de estudos para determinar o quão justos são. No primeiro experimento realizado entre as áreas 1 e 4 (figura 15), notou-se que, durante os 300 segundos de execução da simulação, os fluxos MTCP e TCP compartilharam a mesma largura de banda em aproximadamente 280 Kbytes/s.

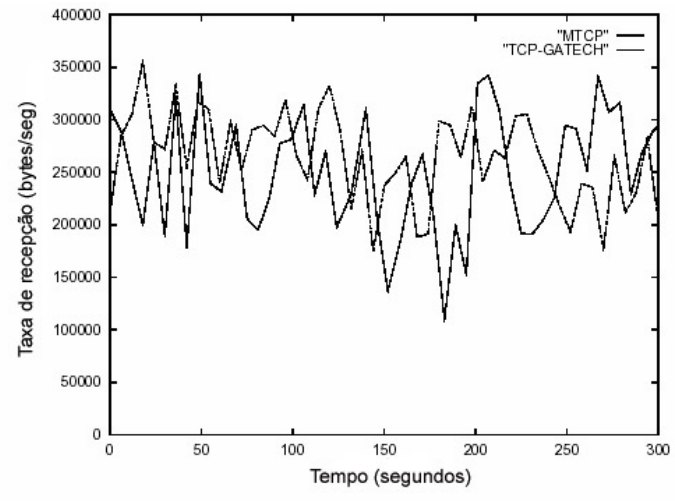


Figura 15. Taxas dos fluxos MTCP e TCP nas áreas 1 e 4. [ROUSKAS *et al*, 1999]

Em outro experimento, envolvendo todas as áreas da árvore (figura 16), percebeu-se a existência de um gargalo entre as áreas 4 e 5, onde a taxa de recebimento mínima da conexão TCP ficou em 60 Kbytes/s. A conexão MTCP acompanhou a taxa TCP, durante os 800 segundos de teste. Isso mostra que o MTCP não utiliza uma banda maior do que uma conexão TCP usa em uma situação de gargalo de um determinado enlace, sendo capaz de ajustar sua taxa, de acordo com a largura de banda disponível. Além do mais, o MTCP reage a cada instante de congestionamento na árvore, enquanto a conexão TCP reage ao congestionamento, somente entre dois pontos.

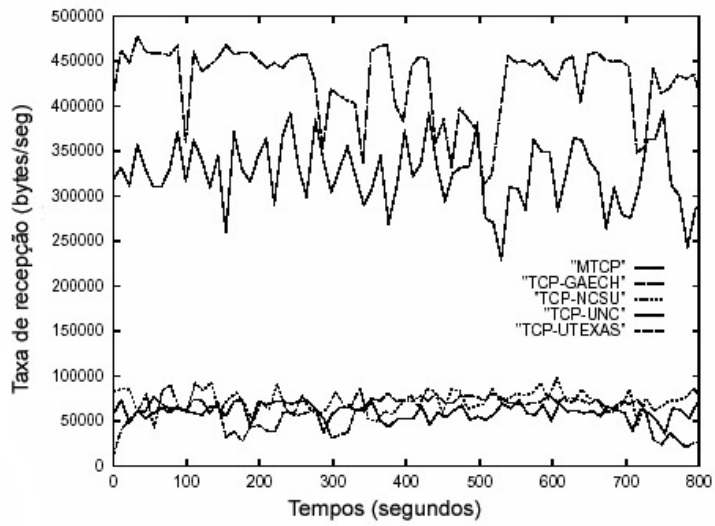


Figura 16. Taxas dos fluxos MTCP e TCP envolvendo todas as áreas. [ROUSKAS *et al*, 1999]

Outro experimento (figura 17) foi realizado, e nele 3 conexões TCP são iniciadas entre as áreas 1 e 4. Enquanto uma transmissão MTCP estava em andamento, as conexões TCP1, TCP2, TCP3 foram iniciadas em diferentes tempos, respectivamente 150, 300 e 410 segundos.Os resultados obtidos foram:

- Quando a primeira conexão TCP1 é iniciada, a transmissão MTCP reduz sua taxa inicial que era de 400 Kbytes/s para 300 Kbytes/s.
- Iniciando a segunda conexão TCP2, o MTCP tende a acompanhá-la em um taxa em torno de 280 Kbytes/s e a conexão TCP1 também reduz sua taxa.
- Por fim, quando a terceira conexão é iniciada, as transmissões MTCP e TCP2 diminuem suavemente suas taxas.

Quando, sucessivamente, as transmissões TCP1, TCP2 e TCP3 vão sendo encerradas, o MTCP vai restaurando sua taxa rapidamente, até chegar novamente em 400 Kbytes/s, mostrando que sua taxa é ajustada sempre de acordo com a largura de banda disponível de um gargalo em uma árvore lógica.

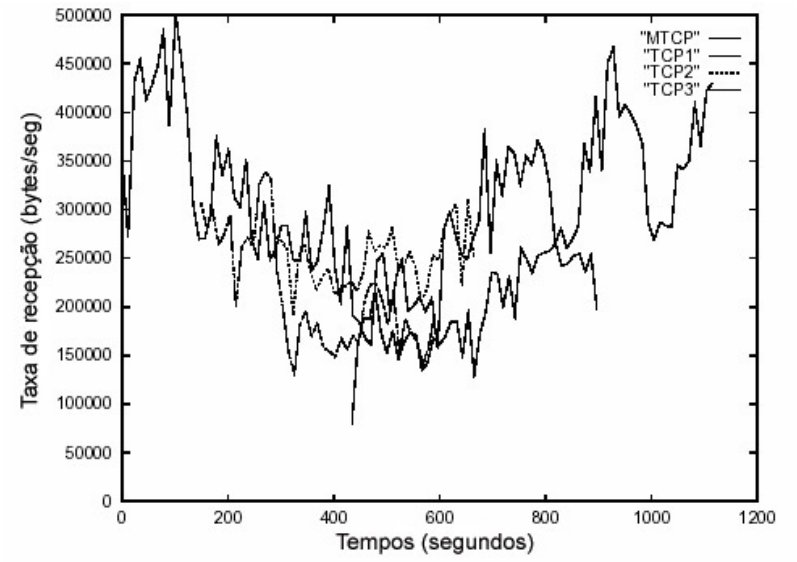


Figura 17. Taxa com múltiplas conexões TCP e 1 MTCP nas áreas 1 e 4. [ROUSKAS *et al*, 1999]

4.2.3 O protocolo "*Nominee-Based Congestion Avoidance*"

O desenvolvimento de um protocolo de controle de congestionamento para um grupo *multicast* muito amplo é um grande desafio, pois diferentes condições de congestionamento em uma árvore *multicast* acontecem, devendo ser levadas em conta na hora de projetar um protocolo, para garantir que ele seja justo no compartilhamento da largura de banda disponível.

Além da **escalabilidade,** outros objetivos devem ser alcançados no desenvolvimento de qualquer protocolo *multicast* como, por exemplo: prevenção contra geração de *feedback* em excesso, causado quando se tem um grande número de receptores; uma eficiente distribuição de pacotes perdidos e um controle de congestionamento justo no compartilhamento da largura de banda entre sessões *multicast* e *unicast.*

Visando a tais objetivos, o protocolo de controle de congestionamento NCA (*Nominee-Based Congestion Avoidance*) [KASERA *et al*, 2000] foi desenvolvido de forma a ajustar a taxa de transmissão no transmissor, de acordo com o recebimento do indicador de perdas de pacotes de apenas um receptor de um grupo *multicast*. O receptor que tiver a conexão de rede mais lenta é nomeado, e passa a ser o representante daquele grupo.

Para determinar qual dos caminhos *multicast* apresenta a maior lentidão, o protocolo NCA utiliza a função:

$g(p,T) = T * \sqrt{p}$, onde *p* é o calculo da probabilidade de perda e *T* é o calculo do RTT do transmissor ao receptor.

Cada receptor envia, periodicamente, seus valores de p e T, diretamente via *unicast* para servidores que são colocados junto aos roteadores, o que possibilita a capacidade de recuperar rapidamente a perda de dados, a partir do ponto de perda, chamados de *repair servers*. Os *repair servers,* baseados no recebimento desses valores através das mensagens de status do congestionamento da rede (CSM), identificam o receptor mais lento através do maior valor da

função *g (p,T)*, o qual repassa através de mensagens CSM para um outro *repair server* mais próximo.

Tendo sido já o receptor identificado e nomeado como o de conexão mais lenta, o transmissor manda, via *unicast*, uma mensagem ao receptor, solicitando o reconhecimento por pacotes o qual se utiliza dessa informação (ACK) para saber em quanto o transmissor irá reajustar sua taxa de transmissão.

A janela de congestionamento no protocolo NCA tem o comportamento similar ao da janela do TCP, no qual o transmissor mantém uma janela de congestionamento *W* e um valor mínimo para o *slow start threshold*. Quando um reconhecimento é recebido, a janela é ajustada da seguinte forma:

Se $(W < threshold) = W \rightarrow W + 1$, **senão** $W \rightarrow W + 1/W$. Ao se detectar uma perda, os valores são ajustados: $threshold \rightarrow W/2$ e $W \rightarrow W/2$ (*congestion avoidance*). É isso mesmo???

A expiração do intervalo de temporização ocorre quando o transmissor não recebe nenhum reconhecimento por certo período de intervalo de tempo que é calculado baseando-se no recebimento do próprio pacote de reconhecimento e também pelo tempo marcado em cada pacote de dados enviado. Nesse caso os valores *W* e *threshold* são reajustados para: $threshold \rightarrow W/2$ e $W \rightarrow 1$ (*slow start*).

O protocolo NCA foi projetado para trabalhar em conjunto com o protocolo de recuperação de perda AER (*Active Error Recovery*). O AER utiliza o conceito de serviços ativos para reduzir o consumo de largura de banda na rede, reduzindo a latência existente na recuperação de perda de pacotes. Os serviços ativos armazenam pacotes (*cache*) nos chamados *repair servers*, que ajudam na distribuição de pacotes perdidos entre o transmissor e os respectivos *repair servers*. Os *repair servers* aproveitam-se, também, do reenvio dos mais recentes pacotes perdidos para agregarem neles (*piggybacking*) informações de reconhecimentos (NACK), ajudando, assim, na redução do uso da banda.

Ao se detectar uma perda, o receptor, após esperar um tempo aleatório, envia um NACK para o *repair server*. Se o *repair server* possuir o pacote pelo qual recebeu o NACK, o pacote é

enviado. Caso não o possua, o NACK é transmitido seguindo a árvore *multicast* acima, até o transmissor, para obter o pacote. Ao receber o NACK de um *repair server,* o transmissor realiza novamente o *multicast* do pacote solicitado para todos os *repair servers* e, finalmente, repassa-o somente para o receptor que solicitou o seu reenvio..

Simulações foram realizadas para demonstrar o comportamento do NCA quando competindo com sessões TCP, focando no compartilhamento justo na largura de banda disponível. O cenário utilizado compreendeu dois nós: em um dos nós colocou-se o transmissor, utilizando o protocolo NCA e, no outro, os receptores interconectados por um único enlace *E* com largura de banda de 1 Mbps e um atraso de propagação de 60 ms.

As simulações da figura 18 foram executadas durante 220 segundos, a primeira sessão TCP1 sendo iniciada em 1,15 segundo. Logo em seguida, aos 10 segundos, iniciou-se a sessão com o protocolo NCA e depois, duas conexões TCP2 e TCP3 aos 21,43 e 21,75 segundos, respectivamente. Observou-se que a sessão NCA e as três sessões TCP receberam, aproximadamente, o mesmo compartilhamento de banda, de 6.000 pacotes por segundo. O protocolo NCA foi capaz de convergir, rapidamente, de maneira justa, para um estado estável, durante a mudança no nível de congestionamento da rede, causado pela introdução do TCP2 e TCP3. Fez-se a mesma simulação, aumentando o número de receptores para 72 e o notou-se o mesmo comportamento.

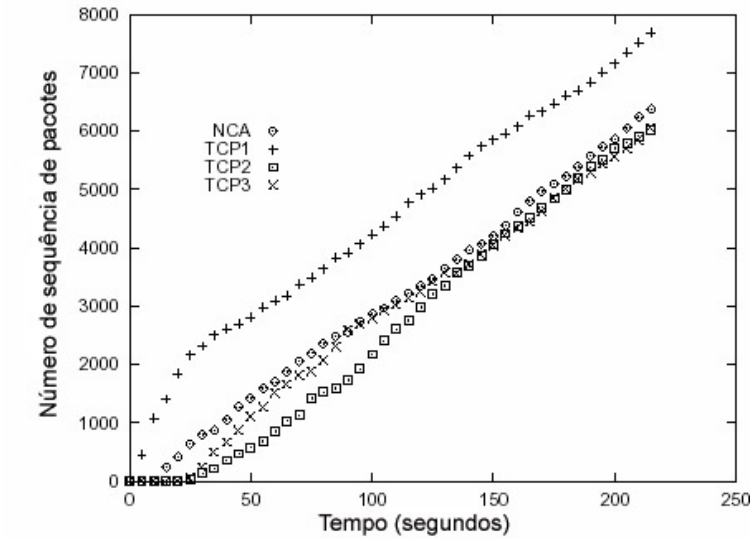


Figura 18. Simulação com 1 sessão NCA e 3 conexões TCP. [KASERA *et al*, 2000]

Em outro experimento de simulação, duas sessões *multicast* sobre o mesmo enlace E foram iniciadas 10 segundos após a primeira, cada uma contendo 2 receptores no nó N2 (figura 19). A simulação mostrou que a taxa de transmissão foi, aproximadamente, de 12.000 pacotes por segundo, a mesma para ambas as sessões, o que indica o compartilhamento de banda justo foi obtido também entre sessões *multicast* NCA.

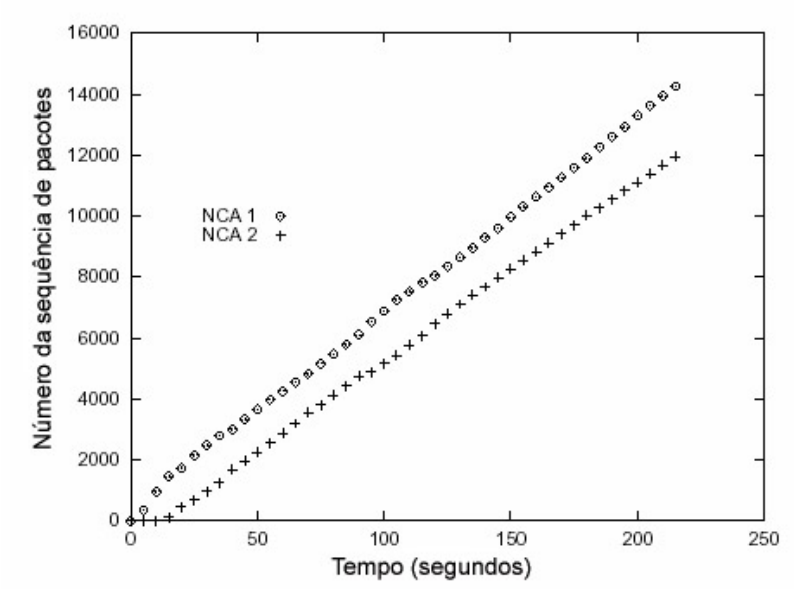


Figura 19. Duas sessões NCA competindo à mesma largura de banda. [KASERA *et al*, 2000]

4.2.4 O protocolo "*Receiver-driven Layered Congestion Control*"

A ausência de mecanismo de controle de congestionamento as aplicações *multicast* podem causar degradação dos recursos da rede, pois um simples fluxo *multicast* pode ser distribuído pela Internet afora, através da criação de uma árvore de *multicast* global.

A proposta do protocolo RLC (*Receiver-driven Layered Congestion Control*) [VICISANO *et al*, 1998] é oferecer uma utilização eficiente da rede de modo amigável ao TCP, adequado para uso em comunicações *multicast*. Através da organização dos dados em camadas é possível que cada camada suporte uma taxa de transmissão variável em grupos separados, deixando que os receptores adaptem a largura de banda disponível, ingressando ou deixando um

ou mais grupos *multicast*, ou seja, recebendo os dados em diferentes taxas, de acordo com a condição da rede.

No TCP, o controle de congestionamento é completamente controlado pelo transmissor, e a única forma que o receptor tem de reduzir a taxa de transmissão é atrasando a geração de pacotes de reconhecimento (ACK), aumentando, assim, o RTT. Já no algoritmo de controle de congestionamento do RLC, os receptores controlam a taxa de transmissão da sessão (*Receiver-driven*), não dependendo de nenhum mecanismo adicional como, por exemplo, o RTT.

O algoritmo utilizado no protocolo RLC tenta deduzir, através dos sinais de congestionamento (perda de pacotes), como a rede está se comportando no momento, fazendo com que cada receptor ingresse ou deixe uma certa camada *i* em função das notificações de congestionamento recebido. A forma como o ingresso ou o abandono das camadas é feito acontece de maneira similar ao comportamento do TCP (AIMD), porém utiliza o esquema DIMD (*Doubling Increase Muliplicative Decrease*) descrito a seguir:

- Se nenhuma perda for detectada durante um período de tempo, aumenta um nível na camada (aumento da taxa em dobro);
- Se houver perda, diminui um nível na camada (diminuição multiplicativa da taxa).

O controle de congestionamento do protocolo RLC não pode trabalhar efetivamente, se os receptores atrás de um mesmo roteador, atuam de uma forma não-coordenada, gerando a perda de pacotes em outros receptores, atuando, dessa forma, de maneira injusta. Um receptor, ao deixar uma camada, não terá efeitos a não ser que todos os receptores que estão compartilhando o mesmo gargalo de enlace deixarem também a camada. A falta desse sincronismo pode gerar o seguinte problema: se um receptor causa um congestionamento pela adição de uma nova camada, outro receptor pode interpretar uma perda como conseqüência disso e abandona uma camada. Esses problemas podem ser minimizados através do uso de pontos de sincronização que correspondem a pacotes especiais nos fluxos de dados.

Um receptor pode ingressar nas camadas somente em pontos de sincronização que são mais freqüentes nas camadas inferiores do que nas camadas superiores. A distância entre os

pontos de sincronização é relacionada com o tempo que um receptor deve gastar e um nível antes do seu ingresso. Após algum tempo, é esperado que os receptores que compartilham o mesmo gargalo devem estar ingressando e abandonando as camadas de forma sincronizada. Para diminuir a probabilidade de que um ingresso falhe, são geradas periodicamente pequenas rajadas de pacotes, antes do ponto de sincronismo, os quais são seguidos de um longo período no qual nenhum pacote é enviado. Isso faz com que a taxa de dados seja dobrada em cada camada e se apenas um receptor não detectar o congestionamento durante a rajada será permitido a ele o ingresso na próxima camada superior.

O abandono de uma camada sempre leva certo tempo para ser finalizado, ocasionando uma falha se houvesse uma tentativa de ingresso em uma camada. Para contornar o problema, após uma perda e, conseqüentemente, a diminuição de um nível na camada, um receptor não reage a uma futura perda por um certo tempo chamado de período surdo, dando ao roteador *multicast* tempo suficiente para responder a um pedido de abandono.

Na simulação realizada (figura 20), verificou-se o comportamento do protocolo RLC em competição com tráfegos TCP, o que aconteceu, conectando-se 2 pares de receptores/transmissores compartilhando o mesmo gargalo de 1 Mbps, iniciando 4 ocorrências do protocolo RLC e uma conexão TCP ao longo do caminho.

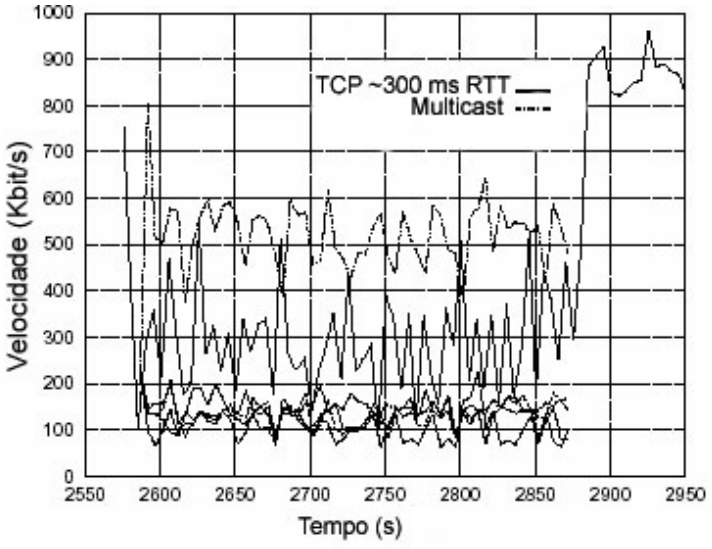


Figura 20. Compartilhamento de banda entre 1 conexão TCP e 4 sessões RLC. [VICISANO *et al*, 1998]

Notou-se que, antes do inicio das ocorrências *multicast*, a conexão TCP utilizava toda a banda disponível no gargalo do enlace e, após o tráfego *multicast* ser iniciado, o rendimento da conexão TCP diminuiu ao nível da banda utilizada pelo tráfego *multicast* de 250 Kbps. O algoritmo utilizado no protocolo RLC alcançou um justo compartilhamento de largura de banda com outras ocorrências do protoloco *multicast* e da conexão TCP.

4.2.5 O protocolo "*Fair Layered Increase/Decrease with Dynamic Layer*"

O protocolo FLID-DL (*Fair Layered Increase/Decrease with Dynamic Layer*) [BYERS *et al*, 2000] introduz dois novos conceitos no desenvolvimento de protocolos de controle de congestionamento amigável ao TCP: o gerenciamento dinâmico das camadas (DL) a fim de reduzir a latência nos processos de ingresso e abandono da camada, e um mecanismo de aumento de taxa muito mais suave do que o utilizado no protocolo RLC.

Quando um *host* deseja parar de receber dados de um grupo *multicast* ele envia uma mensagem IGMP (*Internet Group Membership Protocol*) [FENNER, 1997] de abandono para o roteador, cuja latência até o processo se completar tem a duração de 9 segundos. Durante esse período, os fluxos *multicast* ainda continuam através do roteador, atrasando um ajuste na largura de banda e causando a persistência do congestionamento na rede. Esse problema é conhecido como latência de abandono do IGMP.

O novo mecanismo criado de camadas dinâmicas (DL) vem ajudar no problema de latência de abandono do IGMP. Os receptores controlam sua taxa de recepção, ingressando em um certo número de camadas. Então, para manter a taxa de recepção, os receptores têm que ingressar periodicamente em algumas camadas adicionais. Para reduzir a taxa, os receptores simplesmente não ingressam em nenhuma camada adicional, deixando, automaticamente, que a taxa de transmissão diminua com o passar do tempo, não afetando a resposta a um congestionamento na rede.

O estudo do algoritmo de controle de congestionamento do FLID-DL é dividido nas seguintes partes:

Taxas em diferentes camadas: para iniciar o recebimento de dados de uma sessão *multicast* o *host* ingressa em uma camada básica, que geralmente é a primeira camada, utilizando os três métodos abaixo para a escolha de diferentes taxas nas camadas:

- Método de taxas iguais: taxas iguais para todos os níveis de camadas;
- Método de taxas em dobro: para cada ingresso em um nível de camada, a taxa é dobrada;
- Método de taxas multiplicativas: a taxa em cada camada é proporcional a e^i, onde $e > 1$ é uma constante, e i é o nível.

Dois fatores que devem ser considerados são: o número de camadas necessárias para dimensionar uma certa taxa no receptor, e os ajustes finos na taxa. Um valor grande do número de camadas proporciona mais ajustes finos a serem feitos em uma mudança de taxa, o que leva a uma reação mais suave do mecanismo de controle de congestionamento.

Regras de aumento e de diminuição: O FLID divide o tempo em fatias com duração de T segundos cada. As fatias de tempo são utilizadas para coordenar as atividades de um receptor. Uma nova fatia de tempo inicia-se quando o primeiro pacote é recebido.Os pacotes transmitidos em cada fatia de tempo possuem um índice dessa fatia de tempo. Caso o receptor perceba qualquer perda de pacotes, o receptor deverá reduzir seu nível em 1, ignorando todos os pacotes seguintes.

As fatias de tempo também são utilizadas para coordenar o aumento do nível nos receptores, marcando cada pacote com um sinal de aumento de nível. Esse sinal é utilizado pelos receptores, de acordo com a seguinte regra: se o sinal de aumento de nível da fatia de tempo for maior ou igual que o nível atual i e não existir nenhum pacote perdido, então aumente o nível em 1 no início da próxima fatia de tempo. Assim, se um receptor adicionar uma nova camada e nenhum congestionamento for detectado, os outros receptores atrás do mesmo gargalo com um nível menor poderão também adicionar uma nova camada, aproveitando totalmente a banda disponível. Caso a adição de uma camada cause um congestionamento e tenha que voltar ao nível de camada original, os outros receptores que também adicionaram um nível poderão sentir o mesmo congestionamento e retornar ao nível original também.

Existem mais sinais de aumento de níveis nos receptores de níveis menores do que naqueles de níveis maiores, sendo o uso desses sinais uma das considerações mais importantes para o desenvolvimento do FLID, pois, em conjunto com as diferentes taxas nas camadas, determinam um protocolo justo e confiável, quando colocado em competição com o protocolo TCP.

Simulações foram realizadas para comprovar o funcionamento do protocolo FLID-DL. Para demonstrar como os receptores atrás de um mesmo gargalo coordenam os diversos níveis, 100 *hosts* foram colocados para ingressar numa mesma sessão FLID-DL em tempos aleatórios em um nó R2 e os transmissores colocados no nó R1, interligado por um *link* de 4 Mbits/s, conforme Figura 21.

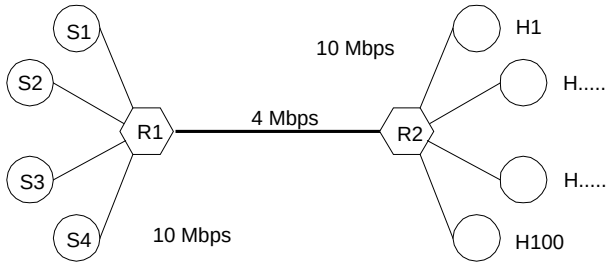


Figura 21. Topologia FLID-DL utilizada na simulação. [BYERS *et al*, 2000]

O resultado obtido na simulação mostrou, de acordo com a figura 22, que todos os 100 *hosts* convergiram para um mesmo nível após 28 segundos de execução do teste.

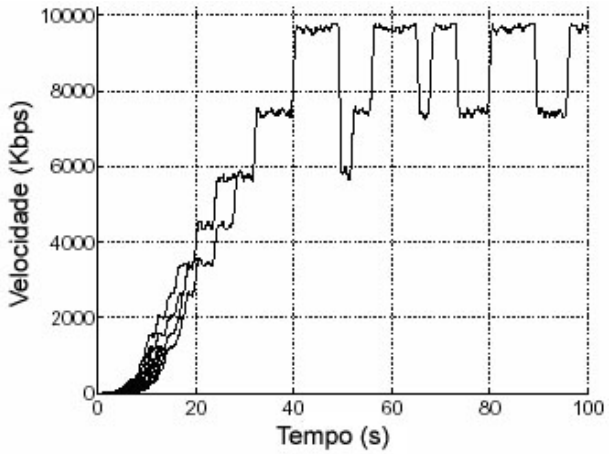


Figura 22. Convergência de 100 hosts atrás do gargalo do enlace. [BYERS *et al*, 2000]

Em um segundo experimento (figura 23), foram colocados em competição *n* fluxos TCP, compartilhando o mesmo gargalo de enlace com *n* sessões FLID-DL. Os roteadores envolvidos na simulação utilizaram o gerenciamento de *buffers Drop Tail*, no qual, a fila estando cheia, os pacotes que chegam vão sendo descartados. Percebeu-se que o FLID-DL é menos justo com fluxos TCP, conforme o tamanho da fila aumenta. A simulação feita com o tamanho da fila de 7 pacotes resultou num compartilhando de largura de banda disponível mais justo do que a com tamanho da fila de 25 pacotes, independente do aumento do número de fluxos durante a execução dos testes.

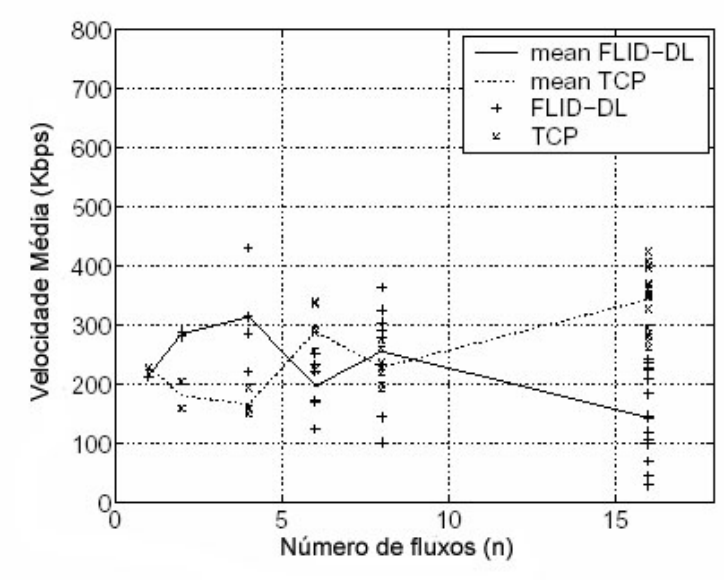


Figura 23. Fluxos TCP e FLID-DL com o tamanho da fila de 7 pacotes. [BYERS *et al*, 2000]

4.2.6 O protocolo "*Multicast enhanced loss-delay based adaptation algorithm*"

O protocolo controle de congestionamento MLDA (*Multicast enhanced loss-delay based adaptation algorithm*) [SISALEM *et al*, 2000b] foi projetado para trabalhar com a arquitetura de transmissão de camadas de dados em vários grupos heterogêneos de *multicast*. Foi concebido, tendo como base o protocolo LDA+, estudado anteriormente. Utiliza também mensagens RTCP para a sinalização entre o receptor e o transmissor.

O MLDA utiliza algoritmo tipo AIMD para adaptar sua taxa de transmissão, mas ao contrário do LDA+, as informações do estado de congestionamento da rede são coletadas pelos receptores. Tais receptores determinam a banda que será compartilhada quando em competição com conexões TCP através de um mesmo caminho, sobre as mesmas condições de perda e atrasos.

Normalmente, o funcionamento de uma estrutura que utiliza a transmissão de dados em camadas determina o tamanho das camadas como sendo fixas. Os transmissores do MLDA recebem, de tempos em tempos, informações sobre a banda compartilhada que é estipulada no receptor, e reajustam o tamanho das diferentes camadas. Assim, os receptores podem determinar o número de camadas a serem ingressados, baseando-se também nas informações dos transmissores. O controle de congestionamento é feito, então, tanto pelo transmissor quanto pelo receptor.

O MLDA trabalha de forma a não precisar de nenhum suporte dos roteadores da rede além da capacidade de encaminhar tráfegos *multicast*. As informações trocadas pelo MLDA e o comportamento do receptor e do transmissor acontecem na seguinte seqüência:

- O transmissor envia periodicamente um relatório com informações sobre as camadas enviadas.
- Após receber o relatório, cada receptor realiza uma medição da perda e o atraso obtido durante a chegada dos dados, em um tempo mínimo necessário para coletar esses dados e determina qual a banda amigável ao TCP a ser utilizada.
- Baseado nesse cálculo o receptor decide se ingressa em uma camada de nível superior (aumento no consumo de banda pelo receptor), se permanece no nível atual ou se a abandona (redução na taxa medida pelo receptor).
- Por fim, o transmissor ajusta o tamanho das diferentes camadas, baseado nos cálculos da largura de banda disponível fornecidos pelos relatórios dos receptores.

A segmentação dos pacotes de dados pertencente à mesma camada possui números de seqüência consecutivos que são utilizados pelo receptor para determinar a taxa de perda em cada camada. No entanto, para diferentes camadas em um fluxo de dados, o receptor necessita de

algum mecanismo para poder organizá-los. Esse controle é feito através da utilização do protocolo RTP o qual especifica, num cabeçalho adicional, informações de seqüência, tipo de dados e o tempo de entrega.

O abandono de uma camada resulta numa redução total da taxa somente se todos os receptores atrás de um mesmo nó de rede deixarem também essa camada. Para que tal resultado seja obtido, é necessária uma certa coordenação entre os diferentes receptores, realizada através de períodos de observação. Somente após esses períodos de observação, determinados pela multiplicação do tempo mínimo necessário para obter o cálculo de perda pelo número de camadas, é que se permite ao receptor ingressar numa camada de nível superior. Isso evita o problema de dois receptores ingressarem em diferentes camadas ao mesmo tempo onde o receptor que ingressou numa camada inferior não ache que a perda observada seja por causa do aumento da taxa, mas sim, por causa do ingresso de um outro receptor numa camada superior.

Para manter o protocolo funcionando de uma maneira confiável, os receptores dependem também de informações enviadas pelo transmissor. Em situações de congestionamento, essa informação pode ser perdida, fazendo com que o receptor não inicie sua adaptação na taxa e entre num estado de *deadlock*. Para prevenir isso, se o tempo calculado pelo envio entre dois relatórios pelo transmissor se esgotar (*timeout*), segundos após os receptores iniciam um período de observação para que, então, o receptor possa ingressar ou deixar a camada.

A simulação realizada para esse protocolo consistiu em 6 receptores conectados ao transmissor através de roteadores de diversas capacidades, utilizando a implementação RED. Cada roteador compartilha fluxos MLDA (*n* receptores) e *m* conexões TCP, sempre enviando dados durante todo o tempo de simulação através de FTP, conforme ilustra a figura 24.

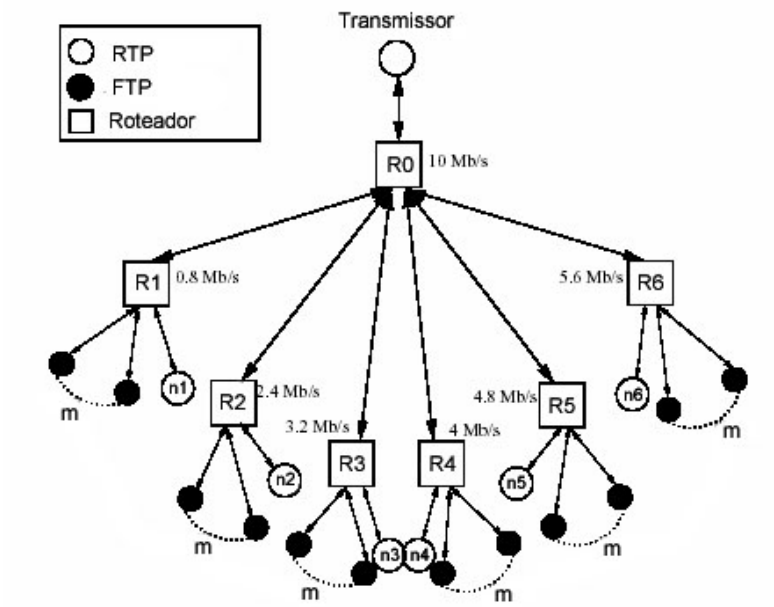
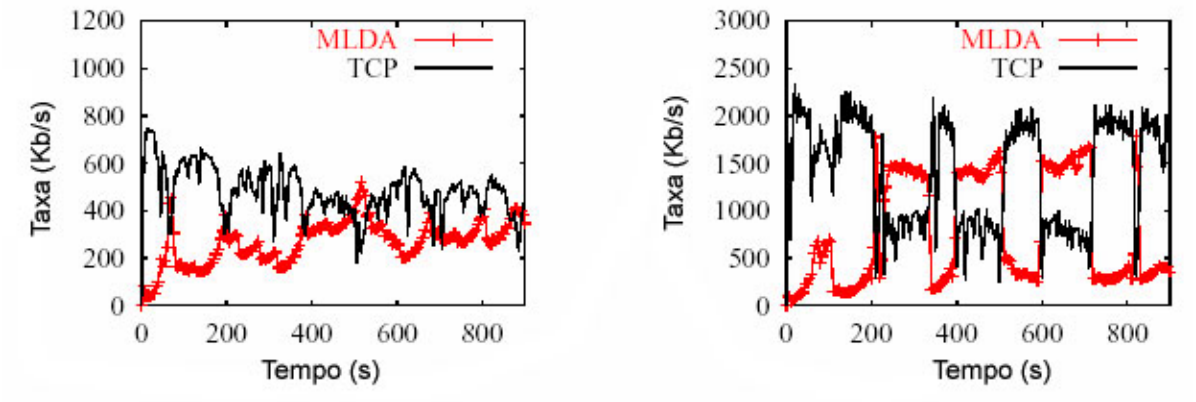


Figura 24. Topologia multicast MLDA utilizada na simulação. [SISALEM *et al*, 2000b]

Após as simulações, os resultados apontaram um compartilhamento justo do protocolo MLDA, quando colocado em competição com conexões TCP, satisfazendo a capacidade de lidar com diferentes receptores.

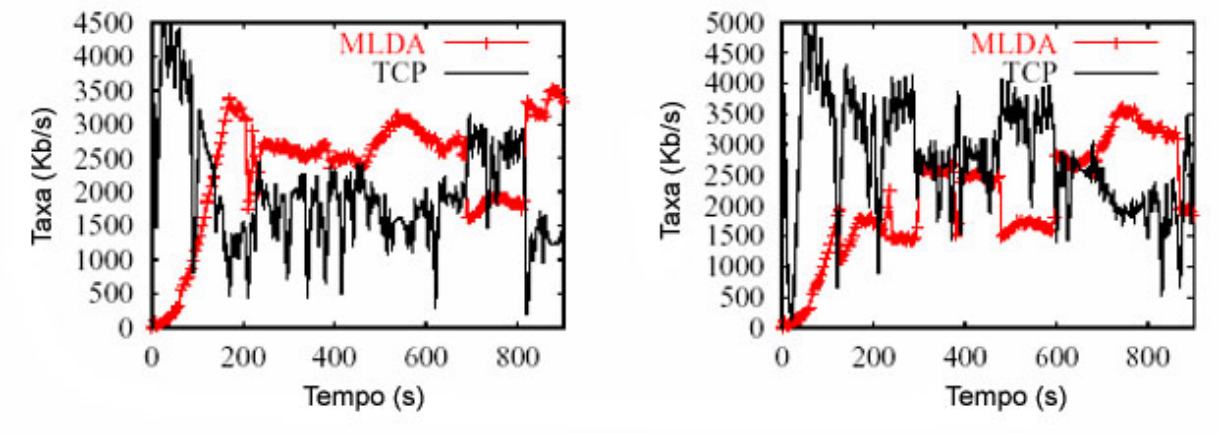
Dependendo da escolha e do número de camadas, temporariamente os receptores cuja largura de banda é menor (n1 e n2), apresentou-se de maneira injusta, como mostra o resultado da simulação ilustrada pela figura 25.



a) Roteador R1 b) Roteador R2

Figura 25 Distribuição de banda entre TCP e MLDA nos roteadores R1 e R2. [SISALEM *et al*, 2000]

Nos demais receptores cuja largura de banda é maior (figura 26), notou-se uma maior estabilidade na distribuição da largura de banda amigável ao TCP. A melhora nos resultados para tornar a distribuição mais justa, com oscilações menores, requer o uso de um grande número de camadas o que aumenta a complexidade do protocolo em gerenciar e reorganizar as camadas.



c) Roteador R5 d) Roteador R6

Figura 26 Distribuição de banda entre TCP e MLDA nos roteadores R5 e R6. [SISALEM *et al*, 2000]

A Tabela 2 mostra a razão entre a largura de banda compartilhada no MLDA e a largura de banda compartilhada entre (*m)* conexões TCP, quando colocado em competição em cada roteador.

Fluxo (m)	R1	R2	R3	R4	R5	R6
1	0,66	0,7	1,03	1,04	1,5	0,95
8	0,5	0,9	0,85	1,09	0,92	1,02

Tabela 2. Razão da largura de banda compartilhada entre MLDA e (m) conexões TCP.

4.2.7 O protocolo “Rainbow”

O Rainbow (*Reliable multicast by individual bandwidth adaptation using window*) [YANO *et al*, 2000a] foi projetado para transmitir uma grande quantidade de dados a um grande número de receptores em ambientes heterogêneos, de maneira eficiente e confiável, em que cada

receptor mantém sua própria janela de congestionamento e, individualmente, executa seu controle de prevenção de congestionamento. Baseado no modelo de serviço BCFS (*Breadcrumb Forwarding Services*) o Rainbow coexiste de maneira amigável com o TCP, conseguindo um alto aproveitamento da rede, em cada enlace de uma sessão *multicast*.

O modelo BCFS [YANO *et al*, 2000b] tem como característica principal o rápido estabelecimento e encerramento de grupos através de pedidos agrupados na troca de dados, de forma análoga ao diálogo feito no TCP para estabelecimento e encerramento de conexões. Para identificar os pedidos de uma mesma origem, o BCFS utiliza rótulos indicados pelo par (Origem/Rótulo), servindo, também, para diferenciar os caminhos percorridos pelos rótulos em uso. O BCFS inclui, ainda, um esquema de numeração de nível, no cabeçalho do pacote, para aumentar o controle do serviço de transporte e, seletivamente, encerrar o estado de "*breadcrumb*".

A seqüência de eventos que acontecem no BCFS é a seguinte:

1) ***Request****:* um receptor envia um pacote de solicitação com o rótulo e o número do nível (inicio do estado "*breadcrumb*").
2) ***Setup:*** um roteador que recebeu a mensagem de solicitação mantém o estado e o nível associado ao rótulo somente, sem armazenar a cópia da mensagem.
3) ***Suppression:*** o roteador encaminha a mensagem de solicitação árvore acima até a raiz, se o rótulo para aquele roteador for novo ou se número do nível for maior que o nível mais alto armazenado no roteador.
4) ***Reply:*** o transmissor, em resposta à mensagem de solicitação, envia os dados solicitados junto com o rótulo da mensagem de solicitação e o número do nível a ser encerrado.
5) ***Forwarding:*** o roteador responde a mensagem diretamente ao enlace, ao qual o rotulo é associado.
6) ***Teardown:*** o roteador remove o estado "*breadcrumb*" do encaminhamento do enlace associado ao rótulo, se na mensagem de resposta houver uma numeração de nível maior do que o mantido pelo roteador.

Para fornecer os dados em diferentes taxas para cada receptor, sem degradar as condições da rede, o controle de congestionamento obedece aos seguintes princípios:

Pedido de transmissão e dados agregados: os receptores que possuem o mesmo tamanho de janela usam o mesmo rótulo em um pedido de transmissão (TRQ). Se um receptor enviar um pedido de transmissão após um outro pedido com o mesmo rótulo ter sido enviado e este último chegar antes no roteador, os pedidos de transmissão são agregados e cópias idênticas dos pacotes de dados são enviadas a todos os receptores de mesmo rótulo.

Janela de congestionamento individual: cada receptor, independentemente, executa seu controle de janela de congestionamento. Pacotes que chegam ao receptor causam um aumento na janela. O algoritmo de controle de janelas do Rainbow imita o TCP no sentido de aumentar o tamanho da janela, quando todos os pacotes da janela são recebidos, ou de diminuir a janela pela metade, quando a perda de pacotes é detectada. Um receptor inicia o tamanho de sua janela em $cw = 1$ quando ele envia seu primeiro TRQ. Estando na fase de *slow start* o tamanho da janela é incrementado de 1 a cada recepção do pacote. Ao detectar uma perda, o TRQ é reinicializado e o receptor entra no modo *congestion avoidance,* aumentando o tamanho da janela de congestionamento em 1, quando todos os pacotes da janela atual forem recebidos.

Reconstrução de dados: dois receptores que possuem a mesma largura de banda, provavelmente, terão o mesmo tamanho da janela quando l é esperado que haja uma agregação nos TRQs. No caso de dois receptores, em que um deles possui a metade da largura de banda do outro, o receptor mais lento recebe metade dos dados direcionados para o receptor mais rápido. Um receptor pode, dessa forma, reconstruir os dados originais sem falhas, a partir de parte dos dados, através da injeção na rede de pacotes codificados por uma origem digital, evitando retransmissões desnecessárias na distribuição confiável de uma grande quantidade de dados.

A detecção de perda no Rainbow é baseada na expiração do intervalo de temporização (*timeout*). Cada receptor mede o tempo entre o pedido e a chegada da resposta dos dados e calcula o RTT. Quando um TRQ é enviado, o tempo é marcado baseado no cálculo do RTT e, se não houver resposta indicando que os pacotes de dados chegaram no tempo determinado, o

pacote é considerado perdido e o tamanho da janela é diminuído pela metade, assim como no TCP.

Para explorar o comportamento do protoloco Rainbow, experimentos foram realizados, utilizando-se o simulador de rede NS2, com 5 receptores compartilhando o mesmo enlace de 400 Kbps a partir de diferentes larguras de banda, utilizando a implementação RED com pacotes de tamanho de 512 bytes conforme figura 27.

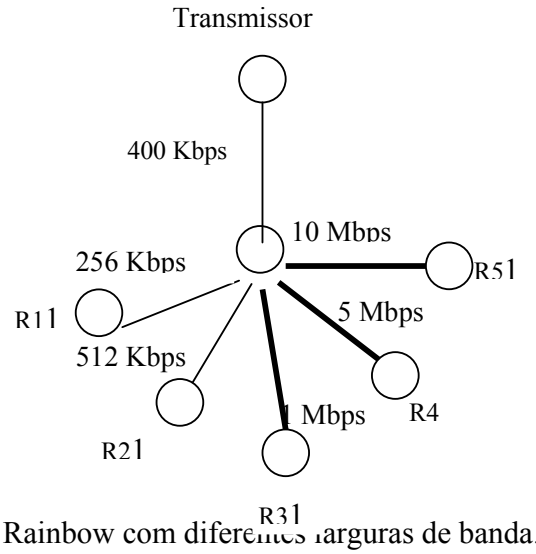


Figura 27. Topologia Rainbow com diferentes larguras de banda. [YANO *et al*, 2000a]

Observou-se, de acordo com os resultados do experimento ilustrado pela figura 28, que todos os receptores receberam uma taxa apropriada conforme cada largura de banda disponível. Os 4 receptores com largura de banda maior (R2,R3,R4,R5) receberam uma taxa em torno de 350 Kbps e o receptor mais lento (R1) recebeu uma taxa um pouco acima de 200 Kbps. Todos os receptores iniciaram suas sessões em tempos aleatórios e os resultados foram medidos através do valor médio de 100 execuções.

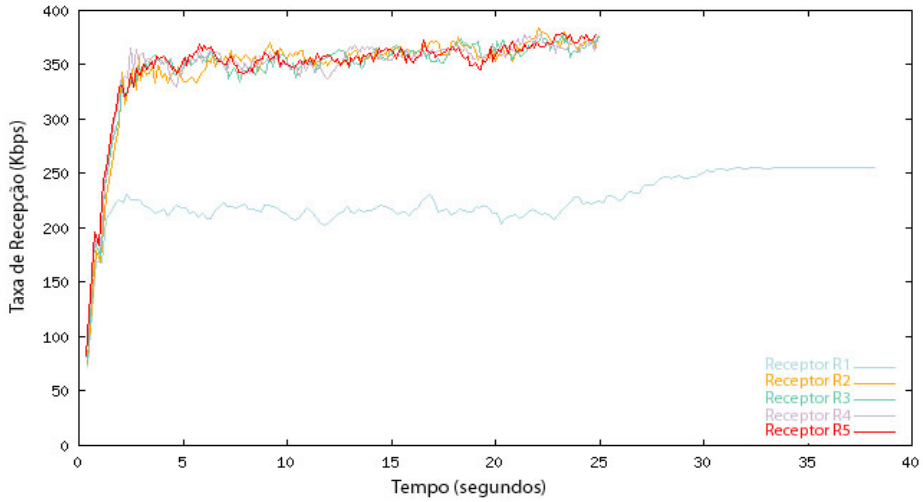


Figura 28. Taxa obtida nos 5 receptores da simulação Rainbow. [YANO *et al*, 2000a]

Experimentos com dois receptores, um utilizando sessões *multicast* Rainbow e o outro com fluxos TCP, foram realizados compartilhando um mesmo gargalo de um enlace, que varia entre 128 Kbps e 512 Kbps. A razão entre a taxa de recepção do Rainbow e o TCP, ao longo do teste, ficou estável em 1,05 em toda a variação da largura de banda, apresentando um comportamento justo, devido ao Rainbow apresentar uma similaridade no seu algoritmo de adaptação de taxa com a do TCP.

Comparou-se o grau de eficiência do protocolo Rainbow (figura 29) repetindo-se a mesma simulação anterior. Substitui-se a sessão *multicast* Rainbow pelo o protocolo RLC, que se comportou nos casos de largura de banda menor de maneira injusta, ocupando 40 % a mais de banda compartilhada que o TCP em relação ao protocolo Rainbow.

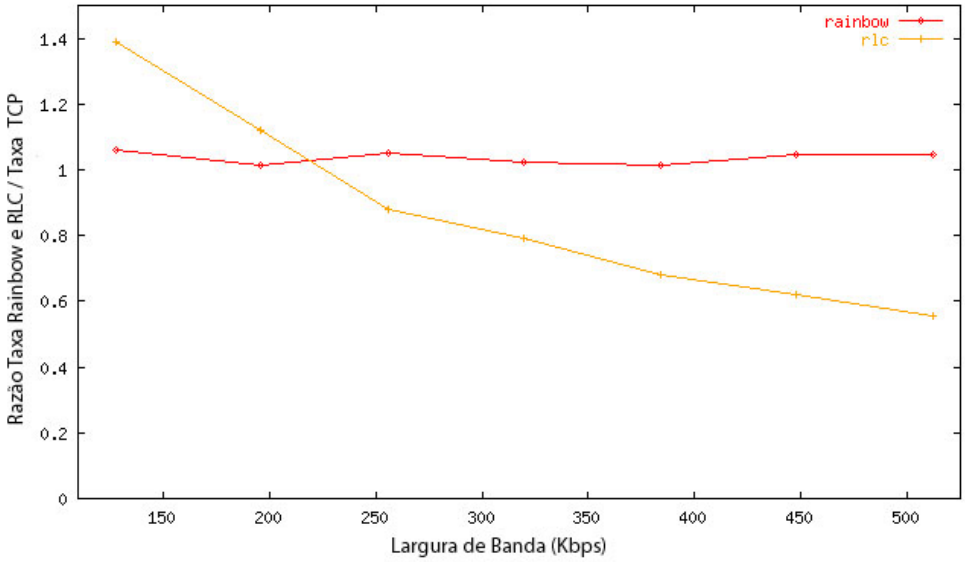


Figura 29. Comparação entre o Rainbow e o RLC. [YANO *et al*, 2000a]

5. Uma comparação entre os protocolos

Definir qual protocolo é o mais adequado para realizar uma determinada tarefa depende, na maioria das vezes, das características da rede e dos requisitos de tráfegos de uma aplicação. Projetar um protocolo a ser utilizado em um ambiente global como a Internet consome muito tempo e dinheiro e somente é aplicado (o protocolo) quando oferece um grande aumento no desempenho em relação à estrutura que se tinha anteriormente. A mesma consideração é feita, quando analisamos a complexidade de um protocolo, cuja melhora do desempenho e seu grau de justiça devem, suficientemente, o *overhead* adicional.

De acordo com os estudos realizados no capítulo 4, foi possível a criação de uma tabela comparativa (tabela 3) que mostra as características principais dos protocolos apresentados no presente trabalho. A terceira coluna mostra o tipo do mecanismo de controle de congestionamento, se é baseado em janela ou em taxa. A coluna seguinte mostra os protocolos que necessitam de um suporte adicional da rede. Protocolos fim-a-fim não necessitam de um

suporte adicional, pois podem ser completamente implementados no nó final. A complexidade de um protocolo é apresentada em uma coluna, levando-se em conta, também, a complexidade exigida no gerenciamento de dados nas camadas.

Para saber se um protocolo é indicado para o uso em aplicações que necessitam de uma taxa estável, uma coluna foi criada para classificá-los quanto à oscilação na taxa, comportando-se de uma maneira suave (*smooth*) ou dente de serra (*sawtooth*). A variação na taxa de transmissão em protocolos baseados em camadas depende do número de camadas utilizadas na distribuição da largura de banda. A classificação amigável ao TCP dos protocolos é baseada, levando-se em conta se o aumento da taxa de transmissão dos protocolos amigáveis ao TCP é mais agressiva que o aumento da taxa das conexões TCP. O nível de classificação como sendo bom, indica que não há sinais de um comportamento injusto contra o TCP. O nível aceitável indica que o protocolo apresenta um bom comportamento amigável ao TCP em geral, mas apresenta problemas de justiça em alguns casos. O nível limitado é dado aos protocolos que apresentam sinais claros de um comportamento injusto contra o TCP em algumas condições da rede, por exemplo, altas taxas de perdas.

Protocolo	Unicast / Multicast	Mecanismo de Controle de Congestionamento	Auxílio da Rede	Complexidade do Protocolo	Oscilação da Taxa	Taxa de Transmissão	Amigável ao TCP
LDA+	Unicast	Taxa	Fim-a-Fim	Alta	Dente de serra	Única	Aceitável
RAP	Unicast	Taxa	Fim-a-Fim	Baixa	Dente de serra	Única	Limitado
TEAR	Unicast	Taxa	Fim-a-Fim	Baixa	Suave	Única	Bom
TFRC	Unicast	Taxa	Fim-a-Fim	Média	Suave	Única	Bom
RLA	Multicast	Janela	Fim-a-Fim	Baixa	Dente de serra	Única	Bom
MTCP	Multicast	Janela	Fim-a-Fim	Baixa	Dente de serra	Única	Bom
NCA	Multicast	Janela	Obrigatório	Baixa	Dente de serra	Única	Bom
RLC	Multicast	Taxa	Opcional	Média	Suave	Variável	Bom
FLID-DL	Multicast	Taxa	Fim-a-Fim	Média	Depende da Camada	Variável	Aceitável
MLDA	Multicast	Taxa	Fim-a-Fim	Média	Depende da Camada	Variável	Aceitável
Rainbow	Multicast	Janela	Fim-a-Fim	Alta	Depende da Camada	Variável	Aceitável

Tabela 3. Tabela comparativa das características dos protocolos.

Alguns protocolos baseados em taxa, utilizando o algoritmo AIMD, apresentam menor complexidade. Uma exceção é o LDA+ que utiliza o protocolo RTP para a troca de informações de *feedback*, não necessitando nenhum outro novo mecanismo para a troca de informações de perdas e atrasos entre o transmissor e o receptor. Isso faz com que o protocolo se torne complexo, mesmo gerando informações de *feedback* em menor escala, se comparado com o protocolo RAP.

O Protocolo RAP recebeu a classificação limitada no comportamento amigável ao TCP, pois se observou nos experimentos de simulação certo grau de injustiça quando em competição com certas implementações do TCP, visto no item 4.1.2.

Tanto o protocolo TEAR como o TFRC apresentam uma excelente suavidade na oscilação da taxa de transmissão e um bom comportamento amigável ao TCP. Essas características são desejáveis em uma transmissão de áudio e vídeo quando a estabilidade da taxa de transmissão é relativamente importante.

O protocolo NCA foi classificado como obrigatório em relação à necessidade de um suporte adicional da rede, pois o mesmo se utiliza do conceito de serviços ativos, visto no item 4.2.3, localizado em servidores ao lado de cada roteador na rede. Já a utilização de um auxílio da rede é classificada como opcional no protocolo RLC, pois não depende de nenhuma informação adicional como, por exemplo, o RTT, em seu controle de congestionamento.

Dentre os protocolos que envolvem um grande número de receptores, o RLC apresentou a oscilação da taxa suave, sem depender totalmente da quantidade de camadas envolvidas, mas de acordo com as condições da rede. Apresentando uma característica diferente, os protocolos FLID-DL, MLDA e Rainbow são totalmente dependentes do número de camadas a serem ingressadas ou abandonadas para determinar o grau de oscilação na taxa de transmissão.

6. Conclusões

Neste trabalho, discutiu-se a necessidade de mecanismos de controle de congestionamento para que os fluxos UDP se comportem de forma amigável ao TCP. Os protocolos de controle de congestionamento amigável ao TCP foram apresentados e cada protocolo mostrou soluções para tentar evitar um colapso da rede.

No protocolo FLID-DL, a solução criada para o problema de latência de abandono do IGMP contribuiu para o desenvolvimento de um mecanismo de aumento de taxa muito mais suave. O uso de ajustes finos na definição de uma taxa de transmissão amigável ao TCP pelo protocolo RAP é uma característica que se mostrou eficiente para melhorar o grau de justiça, durante as conexões quando o TCP é vulnerável a múltiplas perdas. A idéia de utilizar serviços ativos pelo protocolo NCA contribuiu para o aumento do desempenho em termos de redução do uso da largura de banda na rede.

Nas simulações reais, sem o uso do simulador de rede NS2, as considerações sobre segurança do receptor ou transmissor não foram levadas em conta. A criação de mecanismos de autenticação, incorporados nos protocolos amigáveis ao TCP, é necessária para garantir que as informações de *feedback* sejam aceitas somente pelo receptor ou transmissor participante da transmissão. Isso evita que os mecanismos de controle de congestionamento sejam explorados através da inserção de informações falsas.

Como trabalho futuro, pretende-se investigar como os protocolos de controle de congestionamento amigáveis ao TCP se comportam em cenários heterogêneos, com pacotes de dados de curta duração, ao invés de um fluxo contínuo de dados. Além disso, sugere-se desenvolver um método-padrão de comparação entre os protocolos de congestionamento amigáveis ao TCP, baseados em situações reais.

Referências Bibliográficas:

[BYERS *et al*, 2000] BYERS J.; FRUMIN, M.; HORN, G.; LUBY, M.; ROETTER, A.; SHAVER, W. FLID-DL: Congestion Control for Layered Multicast. Palo Alto, CA, **Proc. 2nd Int'l Wkshp. Networked Group Common**, Nov. 2000.

[COMER 1997] Douglas E. Comer, Internetworking with TCP/IP, Volume 1: Principles Protocols, and Architecture, 3rd Edition, Prentice Hall, 1997.

[FENNER 1997] FENNER W. Internet Group Management Protocol version 2. IETF, **RFC 2236**, Jan. 1997.

[FLOYD *et al*, 1999] FLOYD, S.; FALL, K. Promoting the Use of End-to-End Congestion Control in the Internet. **IEEE/ACM Transactions on Networking**, Aug. 1999.

[FLOYD 2000] FLOYD, S. Congestion Control Principles, **RFC 2914**, Best Current Practice, Sep. 2000.

[HANDLEY *et al*, 2003] HANDLEY, M.; PAHDYE, J.; FLOYD, S.; WIDMER J. TCP Friendly Rate Control (TFRC). **RFC 3448**, Standard Track, Jan 2003.

[HASEGAWA *et al*, 2001] HASEGAWA, G.; MURATA M. Survey on fairness issues in TCP congestion control mechanisms. **IEICE Trans. Commun**., v. E84-B, no. 6, p. 1461-1472, Jun. 2001.

[JACOBSON 1990] JACOBSON V. Modified TCP congestion avoidance algorithm. **Note sent to end2end-interest mailing list**, Apr. 30, 1990.

[JACOBSON *et al*, 1993] JACOBSON V.; FLOYD S. Random early detection gateways for congestion avoidance. **IEEE/ACM Transactions on Networking**, Aug. 1993.

[KASERA *et al*, 2000] KASERA, S. K.; BHATTACHARYYA, S.; KEATON, M.; KUROSE, J. Scalable Fair Reliable Multicast Using Active Services. **IEEE Net.** (Special Issue on Multicast), v. 14, no. 1, p. 48–57, Jan./Feb. 2000.

[KUROSE 2002] James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, 1st Edition, Addison Wesley, 2002.

[PADHYE *et al*, 2000] PADHYE, J.; FLOYD, S.; HANDLEY, M.; WIDMER, J. Equation-Based Congestion Control for Unicast Applications. Stockholm, **ACM SIGCOMM 2000**, Aug 2000.

[REALNETWORKS 2000] RealNetworks, Live Broadcast Distribution with RealSystem Server 8, Dec. 2000.

[REJAIE *et al*, 1999] REJAIE, R.; HANDLEY, M.; ESTRIN, D. Rap: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet. **Proc. IEEE INFOCOM**, Mar. 1999.

[RHEE *et al*, 1999] RHEE, I.; ROUSKAS, G.; BALAGURU, N. MTCP: Scalable TCP-Like Congestion Control for Reliable Multicast. **Proc. IEEE INFOCOM**, v. 3, p. 1265–73, Mar. 1999.

[RHEE *et al*, 2000] RHEE, I.; OZDEMIR, V.; YI, Y.; TEAR: TCP Emulation at Receivers - Flow Control for Multimedia Streaming. Tech. rep., **Dept. of Comp. Sci. NCSU**, Apr. 2000.

[SCHULZRINNE *et al*, 1996] SCHULZRINNE, H.; CASNER, S.; FREDERICK, R.; JACOBSON, V. RTP: a transport protocol for real-time applications, Tech. Rep. **RFC 1889**, IEEE Jan. 1996.

[SCHULZRINNE *et al*, 1998] SCHULZRINNE, H.; SISALEM, D. The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme. **Network and Operating System Support for Digital Audio and Video,** Jul. 1998.

[SISALEM *et al*, 2000a] SISALEM, D.; WOLISZ, A. LDA+ TCP-friendly adaptation: A measurement and comparison study. **Proc. International 15 Workshop on Network and Operating Systems Support for Digital Audio and Video**, Jun. 2000.

[SISALEM *et al*, 2000b] SISALEM, D.; WOLISZ, A. MLDA: A TCP-friendly Congestion Control Framework for Heterogeneous Multicast Environments. **8th Int'l. Wksp. QoS**, Jun. 2000.

[STEVENS 2000] W.R. Stevens, TCP/IP Illustrated, Volume1: The Protocols, 16th Printing, Addison Wesley, Feb. 2000.

[VICISANO *et al*, 1998] VICISANO, L.; CROWCROFT J.; RIZZO, L. TCP-like Congestion Control for Layered Multicast Data Transfer. **Proc. IEEE INFOCOM**, v. 3, p. 996–1003, Mar. 1998.

[WANG *et al*, 1998] WANG, H. A.; SCHWARTZ, M.; Achieving Bounded Fairness for Multicast and TCP Traffic in the Internet. **Proc. ACM SIGCOMM**, 1998.

[WIDNER 2000] WIDMER, J. **Equation-Based Congestion Control.** Thesis - University of Mannheim, Department of Mathematics and Computer Science, Feb. 2000.

[WIDNER *et al*, 2001] WIDNER, J.; DENDA, R.; MAUVE, M. A Survey on TCP-Friendly Congestion Control. **IEEE Network**, p. 28–37, May 2001.

[YANO *et al*, 2000a] YANO, K.; McCANNE, S. A Window-based Congestion Control for Reliable Multicast Based on TCP Dynamics. **Proc. ACM Multimedia**, Oct. 2000.

[YANO *et al*, 2000b] YANO, K.; McCANNE, S. The Breadcrumb Forwarding Service: A Synthesis of PGM and Express to Improve and Simplify Global IP Multicast. **ACM SIGCOMM Computer Communication Review**, Apr. 2000.